Microsoft Dynamics™ NAV 5.0

Application Designer's Guide

Руководство разработчика приложений

DRAFT

Черновой вариант перевода, ноябрь 2008 г.

www.microsoft.com/rus/dynamics/nav

ЧАСТЬ 1 АРХИТЕКТУРА 12
ГЛАВА 1. АРХИТЕКТУРА И УСТАНОВКА 12
1.1 Клиент-серверная Среда Microsoft Dynamics NAV 12
Dynamics NAV C/SIDE Клиент 13
Сервер 13
1.2 Установка 13
Чего Следует Избегать при Установке 14
Установочный DVD-Диск 14
Установка Клиента Dynamics NAV C/SIDE 14
Установка Клиентской Части для Работы в Автономном Режиме 15
Установка C/SIDE Database Server в Режиме Клиент-Сервер 15
Установка Опции Microsoft SQL Server для Использования с Dynamics NAV C/SIDE Клиент 17
Установка Нескольких Стандартных Серверов 18
Устранение Проблем при Установке Соединения с Базой Данных 18
Пакеты Обновлений и Обновления Системы Безопасности 19
ГЛАВА 2 ВАРИАНТЫ СЕРВЕРОВ 20
2.1 Различные Варианты Серверов 20
Создание Базы Данных 20
Создание Резервных Копий 30
SIFT 31
Мониторинг Производительности. 32
Прочие Отличия 32
ГЛАВА 3 ОБЕСПЕЧЕНИЕ БЕЗОПАСНОСТИ В СИСТЕМЕ DYNAMICS NAV 34
3.1 Система Безопасности 34
3.2 Функциональные Области и Гранулы 39
ЧАСТЬ 2 ОСНОВЫ 44
ГЛАВА 4 ОСНОВЫ СРЕДЫ C/SIDE 44
4.1 Интерфейс C/SIDE 44
Создание Объектов Приложения 44
Является ли система Dynamics NAV объектно-ориентированной? 46
4.2 Что Такое Приложение C/SIDE? 46
Общие Принципы Среды C/SIDE 46
4.3 Физическая и Логическая Базы Данных 48
ГЛАВА 5 РАЗРАБОТКА ПРИЛОЖЕНИЙ В СРЕДЕ C/SIDE 50
5.1 Введение в Разработку Приложений в Среде C/SIDE 50
Понимание Задачи 50
Проектирование Таблиц 51
Разработка Приложений 53
Рекомендуемая Литература по Проектированию Баз Данных 54
ЧАСТЬ 3 ТАБЛИЦЫ 55
ГЛАВА 6 ТАБЛИЦЫ – БАЗОВЫЕ ПОНЯТИЯ 55
6.1 Что такое таблица? 55
Создание Таблиц 57
DRAFT

6.2 Просмотр и Модификация Свойств 62
Просмотр и Модификация Свойств Таблиц 62
Просмотр и Модификация Свойств Полей 64
6.3 Создание Ключей 67
Как Создавать Первичный Ключ 68
Как Создать Вторичный Ключ 69
Как Ключи Влияют на Производительность 71
Как Хранятся Ключи 71
Просмотр и Модификация Свойств Ключа 72
6.4 Идентификаторы, Типы Данных, Форматы Данных в SQL Server для Dynamics NAV 74
Наименование Идентификаторов 74
Представление Типов Данных Dynamics NAV 74
Совместимость Типов Данных 75
Форматы Данных 76
6.5 Сохранение Таблиц и Просмотр Отсортированных Данных 78
6.6 Специальные Поля Таблиц 81
Что Такое Суммирующие Поля SumIndexFields? 81
Что Такое Вычисляемые Поля FlowFields? 81
Использование Полей FlowFilter в Формулах Расчета Вычисляемого Поля 88
6.7 Разделение Таблиц Между Фирмами 89
ГЛАВА 7 ИЗМЕНЕНИЕ И ПОДДЕРЖКА ТАБЛИЦ 90
7.1 Использование Триггеров Таблиц и Полей 90
7.2 Создание Связей между Таблицами 91
Для Чего Используются Связи? 92
Связи между Таблицами и Свойство TableRelation 92
Создание Базовых Связей между Таблицами 93
Создание Связей между Таблицами с Помощью AssistButton 94
Поддержка Связей между Таблицами в SQL Server 95
7.3 Изменение Таблиц, Содержащих Данные 96
Правила, Используемые при Изменении Таблиц 97
7.4 Связанные Объекты (Linked Objects) 97
Определение Свойства Таблицы Linked Object 97
Требования к Связанным Объектам 98
Изменение Таблиц Dynamics NAV с Определенным Связанным Объектом 100
Подключение к Объектам в Других База Данных или Связанных Серверах 100
ГЛАВА 8 СПЕЦИАЛЬНЫЕ ТАБЛИЦЫ C/SIDE 101
8.1 Что Такое Временная Таблица? 101
8.2 Что Такое Системная Таблица? 102
Системная Таблица User (Пользователь) 103
Системная таблица Member Of (Член) 104
Системная Таблица User Role (Роль Пользователя) 104
Системная Таблица Permission (Права Доступа) 105
Системная Таблица Windows Login (Регистрационное Имя Windows) 105
Системная Таблица Windows Access Control (Контроль Доступа Windows) 106

DRAFT

Системная Таблица Company (Фирма) 106
Системная Таблица Database Key Groups (Группы Ключей БД) 107
8.3 Что Такое Виртуальная Таблица? 108
Когда Используются Виртуальные Таблицы 108
Обзор Виртуальных Таблиц 108
Использование Виртуальных Таблиц 109
ЧАСТЬ 4 ФОРМЫ 126
ГЛАВА 9 ФОРМЫ – БАЗОВЫЕ ПОНЯТИЯ 126
9.1 Что Такое Формы? 126
Что Такое Элементы Управления? 127
Что такое Связанные и Несвязанные Формы и Элементы Управления? 127
Что такое Свойства Формы и Элемента Управления? 127
Что такое Триггеры? 127
9.2 Создание Форм 128
Создание Форм с Помощью Мастера Форм (Form Wizard) 129
Создание Карточной Формы 129
Создание Табличных Форм 133
Создание Формы без Использования Мастера 134
9.3 Сохранение, Компиляция и Запуск Форм 135
Сохранение и Закрытие Формы 136
Компиляция Формы 136
Запуск Формы 136
ГЛАВА 10 РЕДАКТИРОВАНИЕ ФОРМ 137
10.1 Свойства Форм и Элементов Управления 137
Наследование Свойств 138
Свойства Формы 138
Основные Свойства Элементов Управления 139
10.2 Типы Элементов Управления 140
Статические Элементы Управления 140
Элементы Управления, Содержащие Данные 140
Контейнеры 140
Контейнеры, Содержащие Данные 140
Прочие Элементы Управления 141
10.3 Добавление Элементов Управления 141
Toolbox 141
Добавление Текстовой Зоны (Text Box) 142
Добавление Текстовой Зоны без Использования Field Menu 143
Создание Надписей, Которые Показывают Описательный Текст 143
10.4 Выбор, Перемещение и Изменение Элементов Управления 144
Выбор Элемента Управления 144
Множественный Выбор 144
Перемещение Элементов Управления 146
Выравнивание Элементов Управления 146
Изменение Размеров Элементов Управления 147
Изменение Размера Контейнерного Элемента Управления 147
DRAFT

10.5 Инструменты Настройки Элементов Управления ... 147
Использование Инструмента Color ... 147
Использование Инструмента Font ... 148
10.6 Настройка Свойств Элементов Управления ... 148
Изменение Свойств Элементов Управления ... 148
Свойства, Связанные с Отображением Данных ... 149
Свойства, Связанные с Контролем Ввода ... 150
Помощь Пользователю ... 151
10.7 Контейнерные Элементы Управления ... 151
Использования Фреймов (Frame) ... 151
10.8 Как Использовать Элементы Управления в Приложениях ... 153
Отображение Нескольких Строк Текста ... 153
Отображение Рассчитанных Значений ... 153
Представление Набора Опций ... 154
Использование Выключателей (Check Box) для Отображения Значений Типа Booleans ... 159
Создание и Использование Командных Кнопок ... 160
Добавление Фигур (Shapes) и Изображений ... 161
Использование Индикаторов для Отображения Значений ... 163
ГЛАВА 11 РАСШИРЕНИЕ ФУНКЦИОНАЛЬНОСТИ ФОРМ ... 164
11.1 Главные и Сопутствующие Формы ... 164
Создание Главной Формы ... 165
Создание Сопутствующей Формы ... 165
Советы ... 166
11.2 Выбор Значений и Проверка Значений ... 166
Создание Связей между Таблицами ... 168
11.3 Детализация Значений ... 170
Настройка Функции Детализировать (Drill-down) ... 171
11.4 Запуск Других Форм ... 171
Добавление Командной Кнопки ... 171
11.5 Создание Кнопок Меню ... 172
Добавление Кнопок Меню на Форму ... 172
11.6 Триггеры Форм и Элементов Управления ... 175
Обзор Триггеров Форм ... 175
Обзор Триггеров Элементов Управления ... 176
Как Создавать и Изменять Триггеры Формы и Элемента Управления ... 177
11.7 Типы и Характеристики Форм ... 178
Типы Форм и Примеры ... 178
Стандартные Средства Навигации ... 181
ЧАСТЬ 5 ОТЧЕТЫ ... 184
ГЛАВА 12 ОТЧЕТЫ – БАЗОВЫЕ ПОНЯТИЯ ... 184
12.1 Что Такое Отчет? ... 184
Компоненты Отчета ... 184
Логическое и Визуальное Конструирование ... 186
12.2 Что Происходит при Запуске Отчета? ... 188

DRAFT

Запуск Отчета ... 188
12.3 Report Designer ... 190
Report Designer ... 190
Section Designer ... 190
Request Options Form Designer ... 191
12.4 Сохранение, Компиляция и Запуск Отчетов ... 192
Сохранение и Компиляция Отчета ... 192
Запуск Отчета ... 193
ГЛАВА 13 РЕДАКТИРОВАНИЕ ОТЧЕТОВ ... 194
13.1 Свойства Отчетов ... 194
Наследование Свойств ... 195
Свойства Отчета ... 195
Свойства Элементов Данных ... 196
Свойства Разделов ... 197
Свойства Элементов Управления ... 197
13.2 Создание Простого Отчета ... 197
Определение Модели Данных ... 197
Создание Разделов ... 201
13.3 Создание Более Сложного Отчета ... 205
Определение Модели Данных ... 205
Создание Разделов ... 208
ГЛАВА 14 РАСШИРЕНИЕ ФУНКЦИОНАЛЬНОСТИ ОТЧЕТОВ ... 211
14.1 Создание Группировок и Итогов ... 211
Определение Модели Данных ... 211
Связь между Итогами и Разделами ... 213
Создание Разделов ... 214
14.2 Триггеры в Отчетах ... 217
Триггеры Отчетов ... 217
Триггеры Элементов Данных ... 218
Триггеры Разделов ... 218
14.3 Создание Отчета с Использованием Виртуальной таблицы ... 218
Использование Таблицы Date Table ... 218
14.4 Создание Простого Документа ... 232
Определение Модели Данных ... 232
Использование Триггеров ... 235
Создание Разделов ... 236
14.5 Создание Непечатаемого Отчета ... 238
Определение Модели Данных ... 238
Создание Формы Запроса ... 239
Использование Триггеров ... 240
14.6 Типы Отчетов ... 241
Отчеты List ... 241
Отчеты Test ... 241
Отчеты Posting ... 242
Отчеты Transaction ... 242
DRAFT

Другие "Нормальные" Отчеты ... 242
Отчеты Document ... 242
ЧАСТЬ 6 ПРОГРАММНЫЕ МОДУЛИ ... 243
ГЛАВА 15 ПРОГРАММНЫЕ МОДУЛИ – БАЗОВЫЕ ПОНЯТИЯ ... 243
15.1 Что такое Программный Модуль Среды C/SIDE? ... 243
15.2 Создание Программных Модулей ... 244
Использование C/AL Editor ... 244
Определение Переменных, Текстовых Констант и Функций в Программном Модуле ... 245
Использование C/AL Symbol Menu ... 250
Компиляция и Сохранение Программного Модуля ... 251
15.3 Использование Программных Модулей ... 251
Ограничения в Использовании Программных Модулей ... 257
ГЛАВА 16 ВВЕДЕНИЕ В ЯЗЫК C/AL ... 258
16.1 Что Можно Делать с Помощью Языка C/AL? ... 258
16.2 Что Такое Программный Оператор, Выражение и Оператор? ... 258
Что такое Выражение на Языке C/AL? ... 259
Введение в Типы Данных Языка C/AL ... 260
Базовые Типы Данных ... 261
Комплексные Типы Данных ... 263
Создание Массивов Переменных ... 264
16.3 Введение в Выражения на Языке C/AL ... 264
Константы ... 265
Использование Переменных в Языке C/AL ... 265
16.4 Язык Управления C/AL ... 271
Составные Операторы ... 272
Условные Операторы ... 272
Использование Операторов Цикла ... 274
Как Добавлять Комментарии к Вашим Программам ... 278
ГЛАВА 17 ИСПОЛЬЗОВАНИЯ ЯЗЫКА C/AL ... 280
17.1 Обзор ... 280
Где Писать C/AL Код ... 280
Повторное Использование Кода ... 281
17.2 Системные Переменные ... 281
17.3 Обработка Ошибок Выполнения ... 282
17.4 Основные Функции Языка C/AL ... 283
Поиск Записей ... 283
Сортировка и Фильтрация Записей ... 284
Вставка, Изменение и Удаление Записей ... 286
Транзакции ... 289
Работа с Полями ... 289
Сообщения и Диалоговые Окна ... 293
Создание Окна для Отображения Хода Выполнения ... 294
Прочие Сообщения для Пользователя ... 295
Форма Быстрого Выбора Опций ... 297
DRAFT

ГЛАВА 18 ОТЛАДКА C/AL КОДА ... 298
18.1 Что Такое Программная Ошибка? ... 298
Синтаксические Ошибки (Syntax Errors) ... 298
Ошибки Выполнение (Run-time Errors) ... 299
Ошибки в Логике Программы ... 302
18.2 Microsoft Dynamics NAV Debugger ... 302
Общее Описание ... 302
Активация Отладчика ... 303
Хранение Отладочной Информации в Файле FIN.ZUP ... 308
Обзор Горячих Клавиш ... 308
18.3 Инструмент Код Приложения ... 309
Использование Инструмента Код Приложения ... 309
ГЛАВА 19 РАСШИРЕНИЕ ВОЗМОЖНОСТЕЙ ЯЗЫКА C/AL ... 311
19.1 Что Такое COM? ... 311
COM и C/SIDE ... 311
19.2 Использование Технологий COM в Среде C/SIDE ... 311
Unsigned char (VT_UI1), SCODE (VT_ERROR) и SAFEARRAY (VT_ARRAY) ... 313
19.3 Использование C/SIDE как Automation Controller ... 314
Создание Письма в Microsoft Word ... 314
Создание Диаграмм в Microsoft Excel ... 325
19.4 Получение Событий в C/SIDE ... 331
Получение Уведомления о Входящем Документе ... 331
Получение События ... 331
Триггеры Событий ... 335
Ограничения для Триггеров ... 335
Ограничения для Входящих Данных ... 336
Отправка Потока ... 336
Запуск и Тестирование Примера ... 337
19.5 Использование компонентов (Custom Controls) из Среды C/SIDE ... 338
Простой Пример ... 338
Установка и Регистрация Компонента ... 338
Использование Компонента из C/AL ... 339
Объявление Компонентов как Переменных ... 339
Создание Формы ... 342
Добавление Кода на Форму ... 344
Обработка Исключений ... 345
ЧАСТЬ 7 ДАТАПОРТЫ ... 346
ГЛАВА 20 ДАТАПОРТЫ ... 346
20.1 Что такое Датапорт? ... 346
Логическая структура ... 347
Сохранение, Компиляция и Запуск Датапорта ... 350
20.2 Создание Датапортов ... 351
Свойства Датапорта ... 351
Формат Файла ... 353
Свойства Элемента Данных ... 353

DRAFT

Свойства Поля Датапорта ... 355
Триггеры Датапорта ... 356
20.3 Экспорт Данных ... 357
Экспорт – Фиксированный Формат ... 357
Улучшенная Версия ... 361
Экспорт – Переменный Формат ... 363
20.4 Импорт Данных ... 364
Импорт – Фиксированный Формат ... 364
Импорт – Переменный Формат ... 368
Импорт или Экспорт: Динамический Датапорт ... 370
ЧАСТЬ 8 XMLPORT ... 377
ГЛАВА 21 XMLPORT ... 377
21.1 XMLport – Базовые Понятия ... 377
Сохранение, Компиляция и Запуск XMLport ... 377
21.2 Создание XMLport ... 378
XMLport Designer ... 379
21.3 Примеры использования XMLport ... 381
21.4 Проверка Данных ... 386
21.5 XMLport и Business Notifications ... 387
ЧАСТЬ 9 ОБЪЕКТЫ MENUSUITE ... 389
ГЛАВА 22 ОБЪЕКТЫ MENUSUITE ... 389
22.1 MenuSuite – Базовые Понятия ... 389
Создание и Изменение Объектов MenuSuite ... 389
22.2 Настройка MenuSuite ... 390
Создание Пунктов Меню ... 390
Установка Свойств ... 391
Уровень Объекта MenuSuite ... 392
22.3 Экспорт Объекта MenuSuite ... 392
22.4 Обновление Содержимого MenuSuite ... 394
ЧАСТЬ 10 МНОГОЯЗЫКОВАЯ ФУНКЦИОНАЛЬНОСТЬ ... 395
ГЛАВА 23 МНОГОЯЗЫКОВАЯ ФУНКЦИОНАЛЬНОСТЬ ... 395
23.1 Многоязыковая Функциональность ... 395
Определение Текущего Языка Приложения ... 395
Выбор Языка из Интерфейса Пользователя ... 395
Текстовые Константы ... 395
Языковые Модули ... 396
Виртуальная Таблица «Windows Language» ... 398
Закладки ... 398
Поддерживать Виды ... 399
NAV ODBC ... 399
Отображение Текста ... 399
Несколько Языков в Документе ... 399
23.2 Создание Приложений Поддерживающих Несколько Языков ... 400
Свойство Name ... 400
Текстовые константы ... 400
DRAFT

Свойство Caption ... 401
Свойство CaptionML ... 401
Создание Caption ... 401
Date Formula ... 402
23.3 Понимание Базового Языка, Используемого при Написании Кода ... 402
Как Увидеть Одновременно и Caption и Name ... 402
23.4 Диапазоны Идентификаторов для Текстовых Констант ... 404
ЧАСТЬ 11 УГЛУБЛЕННОЕ ИЗУЧЕНИЕ ... 406
ГЛАВА 24 SUMINDEXFIELDS ... 406
24.1 SumIndexFields ... 406
SIFT и C/SIDE Database Server ... 406
ГЛАВА 25 ПРЕОБРАЗОВАНИЕ ТИПОВ ДАННЫХ ... 408
25.1 Преобразование Типов Данных в Выражениях ... 408
25.2 Механизмы Преобразования Типов Данных ... 409
Операторы Сравнения (Relational Operators) ... 409
Логические Операторы (Boolean Operators) ... 410
Арифметические Операторы (Arithmetic Operators) ... 411
Полный Обзор Правил Преобразования Типов Данных ... 413
ГЛАВА 26 НУМЕРАЦИЯ В DYNAMICS NAV ... 416
26.1 Как Работает Сортировка Номеров? ... 416
Принципы Нумерации ... 417
Фильтры ... 417
ГЛАВА 27 СРЕДА C/SIDE В МНОГОПОЛЬЗОВАТЕЛЬСКОМ ОКРУЖЕНИИ ... 418
27.1 Обеспечение Целостности Данных в Многопользовательском Окружении ... 418
Транзакции Записи и Восстановления (Write Transactions and Recovery) ... 418
Непротиворечивость Чтения и Конкуренция (Read Consistency and Concurrency) ... 420
Что Такое Блокировка Таблиц? ... 422
Что Такое Выявление Взаимных Блокировок? ... 422
Есть ли Разница между Commit в языке C/AL и в C? ... 423
27.2 Блокировки в Dynamics NAV – Сравнение Двух Вариантов Серверов ... 424
Оба Варианта Серверов ... 424
Минимизация Взаимных Блокировок ... 425
Блокировки в C/SIDE Database Server ... 425
Блокировки в SQL Server ... 425
Различие Блокировок в Коде ... 426
ГЛАВА 28 ОБЗОР ФУНКЦИОНАЛЬНОСТИ CAPTION CLASS ... 427
28.1 Обзор Функциональности Caption Class - ... 427
LANGUAGE ... 427
CAPTIONEXPR ... 427
Код Функции ... 428
CAPTIONREF ... 429
ГЛАВА 29 ПОДДЕРЖКА СИСТЕМЫ БЕЗОПАСНОСТИ НА УРОВНЕ ЗАПИСИ ... 430
29.1 Система Безопасности на Уровне Записи ... 430
DRAFT

Внедрение Системы Безопасности на Уровне Записи 430
Формы, Отчеты и Датапорты 431
Программный Модуль 431
Руководство по Внедрению Системы Безопасности на Уровне Записи 432
Применение Фильтров Защиты при Учете 432
Фильтры Защиты и Сложные Формы 438
ГЛАВА 30 ПРОИЗВОДИТЕЛЬНОСТЬ 445
30.1 Кэш СУБД (DBMS Cache) 445
30.2 Кэш Дисков (Commit Cache) 446
30.3 Командный Буфер 447
30.4 Ключи, Запросы и Производительность 448
30.5 Функции C/AL и производительность на SQL Server 449
Администрирование Базы Данных, Изменение Объектов и Производительность на SQL Server 450
30.6 Хранимая Процедура Входа для SQL Server Option 450
Создание Хранимой Процедуры 450

DRAFT

ЧАСТЬ 1 АРХИТЕКТУРА

ГЛАВА 1. АРХИТЕКТУРА И УСТАНОВКА

В данной главе будет рассказано об архитектуре Microsoft Dynamics NAV, а также о том, как установить клиента Dynamics NAV, C/SIDE Database Server и SQL Server.

Данная глава состоит из следующих разделов:

- Клиент-серверная среда Microsoft Dynamics NAV
- Установка

1.1 Клиент-серверная Среда Microsoft Dynamics NAV

Microsoft Dynamics NAV представляет собой двухзвенное приложение. Оно состоит из системы управления базами данных (СУБД), которая располагается на сервере, и графического интерфейса, работающего на каждой клиентской системе. Кроме того, клиентская часть системы может быть сконфигурирована для работы в автономном режиме. Это означает, что клиентская часть будет выполнять одновременно роли и сервера, и клиента.

Dynamics NAV может использовать две базы данных: стандартную БД в формате Navision и базу данных в формате SQL Server. При работе в автономном режиме можно использовать встроенную клиентскую СУБД клиента для управления стандартной базой данных в формате Navision. СУБД также поставляется и в серверном варианте, что позволяет нескольким клиентским системам одновременно подключаться к общей базе данных.

SQL Server Option для Dynamics NAV может использоваться в автономном режиме. Для этого необходимо вместе с клиентским ПО установить Microsoft SQL Server Express. Это облегченная версия SQL Server. При установке демонстрационная база будет присоединена к локальному SQL Server и будет открыта при первом запуске Dynamics NAV.

При использовании базы данных в формате SQL Server необходимо установить ПО SQL Server на серверной системе в соответствии с рекомендациями корпорации Microsoft. На каждом клиентском компьютере устанавливается клиент Dynamics NAV.

При работе в режиме SQL Server Option клиент обычно использует для связи с SQL Server протокол TCP/IP, хотя также он может применять подключение через ODBC. При работе с C/SIDE Database Server Option для подключения к серверу клиент обычно использует протокол TCP/IP.

После того, как сервер и клиентские системы сконфигурированы, пользователю не требуется знать о том, как они взаимодействуют между собой: этот процесс полностью скрыт от пользователя.

Сервер приложений NAV Application Server (NAS) — это сервер промежуточного уровня, поддерживающий многозвенную архитектуру. Он позволяет выполнять бизнес-логику без

DRAFT

вмешательства пользователя. При помощи NAV Application Server можно организовать взаимодействие с внешними сервисами.

NAV Application Server выступает в роли клиента по отношению к серверу баз данных и может служить сервером для других сервисов. При запуске NAV Application Server открывает указанную базу данных и исполняет код C/AL в заданном программном модуле.

NAV Application Server взаимодействует с C/SIDE Database Server или с SQL Server тем же способом, что и клиентская часть Dynamics NAV. NAV Application Server поддерживает только механизм Windows-аутентификации и автоматически восстанавливает связь с сервером баз данных в случае проблем с сетью.

Dynamics NAV C/SIDE Клиент

Основное предназначение Dynamics NAV C/SIDE клиента — предоставить пользовательский интерфейс, однако его возможности гораздо шире. При работе в автономном режиме клиент способен напрямую подключаться к файлу стандартной базы данных, минуя сервер. Кроме того, клиент отвечает за исполнение всей бизнес-логики. Он считывает из базы данных объекты и обеспечивает контроль над их функционированием. Основная часть приложения Dynamics NAV работает на индивидуальных клиентских системах.

Сервер

Если клиенты выполняют основную работу, что же выпадает на долю сервера? Стандартное серверное ПО системы Dynamics NAV выполняет следующие задачи:

- контроль количества пользователей, одновременно подключающихся к базе данных;
- контроль доступа к данным посредством механизма блокировки;
- слежение за всеми транзакциями записи и чтения, осуществляемыми каждым пользователем;
- выдача данных по запросу клиента;
- фильтрация на основе ключей (key-based filtering) и вычисление SumIndexFields;
- кэширование данных, которые могут быть затребованы повторно.

Это неполный перечень. Он приведен лишь для того, чтобы дать общее представление о функциях сервера. Microsoft SQL Server также выполняет все эти задачи. Однако стандартный сервер, в отличие от SQL Server, осуществляет слежение за различными версиями одной записи. Речь об этом пойдет в последующих главах.

Связка «клиент - сервер баз данных» представляет собой полностью интегрированное решение.

1.2 Установка

Как было указано ранее, Dynamics NAV может быть установлен, как в режиме Клиент-Сервер, так и для работы в автономном режиме.

Существует несколько способов установки клиентского и серверного ПО. В этой главе мы рассмотрим различные способы установки клиента и сервера. Также будут рассмотрены наиболее часто встречающиеся ошибки.

Для успешной установки необходимо:

- установить ПО с установочного DVD-диска Dynamics NAV;
- установить базу данных пользователя;
- установить клиентскую лицензию;
- протестировать систему, в том числе и соединение между клиентом и сервером.

DRAFT

Чего Следует Избегать при Установке

При конфигурировании системы, где хранится база данных, необходимо соблюдать следующие требования

- не используйте утилиты сжатия данных, такие как DoubleSpace, Stacker или DriveSpace;
- ограничивайте количество сервисов и программ, работающих одновременно, они замедляют обработку данных;
- не используйте Lazy Write и подобные программы кэширования.

  Важно

  Используйте Windows Server 2003 или Windows XP в качестве операционной системы для сервера.

  Используйте Windows Vista, Windows XP или Windows Server 2003 в качестве операционной системы для клиентских компьютеров.

Установочный DVD-Диск

Вставьте установочный DVD-диск в привод DVD дисков. В браузере автоматически откроется HTML-страница. Данная страница содержит ссылки на установочные программы и на документацию к ним. С DVD-диска можно установить:

- Dynamics NAV для работы в автономном режиме.
- Dynamics NAV для работы в автономном режиме, использующем Microsoft SQL Server Express Database Engine.
- C/SIDE Database Server for Microsoft Dynamics NAV для работы в режиме Клиент-Сервер.
- Application Server для Microsoft Dynamics NAV.
- Компоненты Commerce Gateway.
- Business Notification Server для Microsoft Dynamics NAV
- Автоматизированную Систему Сбора Данных (ADCS) для Microsoft Dynamics NAV
- ODBC для Microsoft Dynamics NAV

Для работы в режиме Клиент-Сервер с использованием SQL Server отдельно нужно установить Microsoft SQL Server.

Установка Клиента Dynamics NAV C/SIDE

Щелкните Клиент C/Side для Microsoft Dynamics. Запустится стандартная программа установки программного обеспечения ОС Windows. Следуйте ее указаниям.

При установке клиентского ПО можно указать функции программы, которые необходимо установить:

- **Справка**. Интерактивная справка для Dynamics NAV.
- **Демонстрационная база данных**. Демонстрационная база данных используется при запуске Dynamics NAV в однопользовательском режиме.
- **Резервная копия базы данных**. Резервная копия базы данных используется для создания демонстрационной фирмы в базе данных SQL Server.
- **Интеграция Commerce**. Установка компонентов Commerce Gateway.
- **Business Notification Manager**. Эта функция позволяет автоматически отправлять уведомления сотрудникам и партнерам по бизнесу.

DRAFT

- **Employee Portal**. Установка на компьютер компонентов для Employee Portal.
- **Интеграция с Outlook**. Устанавливает на компьютер компоненты для интеграции Microsoft Dynamics NAV с Outlook.
- **Dynamics NAV Gantt Server**. Этот компонент ActiveX позволяет руководителю производства планировать работу производств с использованием диаграмм Ганта, а также обновлять графики производства в Dynamics NAV.

Файлы на Жестком Диске

После завершения работы программы установки вы можете просмотреть содержимое каталогов, куда было установлено программное обеспечение. Ниже приводится перечень некоторых установленных файлов:

FIN.EXE	Исполняемый файл для запуска клиента Dynamics NAV C/SIDE при работе с C/SIDE Database Server или в автономном режиме.
FINSQL.EXE	Исполняемый файл для запуска клиента Dynamics NAV при работе с SQL Server
FIN.FLF	Файл с лицензией, определяющий функциональные возможности системы на основе приобретенного набора гранул.
CRONUS.FLF	Демонстрационный файл с лицензией для вымышленной компании.
DATABASE.FDB	Файл базы данных, используемый по умолчанию. Он содержит все объекты и данные.
MASTER.CHM	Главный файл справки для приложения. Этот файл расположен в подкаталоге вместе с остальными файлами справки.

Дополнительную информацию по установке клиента см. в руководстве *Installation & System Management:* для выбранного сервера. Данные руководства доступны на установочном DVD-диске.

Установка Клиентской Части для Работы в Автономном Режиме

Наиболее простой является однопользовательская конфигурация, когда клиент работает в автономном режиме. Процедура установки в этом случае предельно проста. Необходимо установить клиентское программное обеспечение на компьютере пользователя. Затем вы должны заменить файл database.fdb базой данных, которая была создана специально для этого пользователя. И наконец, замените файл с лицензией fin.flf, который был установлен вместе с исполняемыми файлами, файлом с клиентской лицензией.

Чтобы установить клиента для SQL Server в автономном режиме, при установке клиента следует дополнительно выполнить установку Microsoft SQL Server Express. При этом будет выполнена установка локального SQL Server и присоединена демонстрационная база данных. При запуске клиента, автоматически откроется демонстрационная база данных.

На сервере можно создать свою собственную базу данных, позже будет описано, как это сделать.

Установка C/SIDE Database Server в Режиме Клиент-Сервер

Эта конфигурация подразумевает наличие одного стандартного сервера и нескольких клиентов. Начинать следует с установки клиентского ПО, сконфигурированного для работы в автономном режиме. Его следует установить на серверном компьютере (то есть на компьютере, на котором будет работать серверная программа) в соответствии с процедурой, описанной выше. Это необходимо сделать в первую очередь, чтобы обеспечить установку базы данных и клиентской лицензии. Единственное отличие состоит в том, что в данном случае может оказаться целесообразным выделить для базы данных отдельный каталог или отдельный диск. В одной из последующих глав мы обсудим вопросы, связанные с файлом базы данных и оптимизацией его настроек.

DRAFT

После установки клиентского ПО на сервере можно запускать программу установки серверного ПО - C/SIDE Database Server. Это отдельная установочная программа.

C/SIDE Database Server устанавливается с помощью стандартной программы установки программного обеспечения ОС Windows. Данная программа проведет вас по всем шагам установки. Если в процессе установки вы затрудняетесь в выборе из предлагаемых опций, воспользуйтесь руководством Installation & System Management: Database Server for the Dynamics NAV C/SIDE Client. Данное руководство содержит пошаговые описания действий, которые нужно выполнить в процессе установки.

> **Важно**
>
> Рекомендуется попрактиковаться в установке C/SIDE Database Server, перед тем как устанавливает его у заказчика.

Программа установки попросит указать место расположения базы данных и файла с лицензией. Выберите базу данных и файл с клиентской лицензией. Программа также создаст сервис и запустит его в конце процедуры установки. Именно этот сервис отвечает за функционирование стандартного сервера Dynamics NAV. После завершения процедуры установки этот сервер будет находиться в рабочем состоянии.

Следующий шаг — установка Dynamics NAV на всех клиентских компьютерах. Процедура похожа на ту, которая используется для установки клиента в автономном режиме, за некоторым исключением. Необходимо установить только клиентское ПО (файл базы данных не нужен, поэтому следует выбрать тип установки Minimum). Базу данных устанавливать не нужно, так как она уже находится на сервере. Однако следует скопировать клиентскую лицензию на каждый клиентский компьютер.

Завершающим шагом после установки клиентского ПО является создание на рабочем столе пользователя ярлыка и его тестирование. Ярлык должен содержать определенные параметры, например:

```
"C:\Program Files\...\fin.exe" servername=MyServer, nettype=TCPS
```

В этом примере **servername** задает имя C/SIDE Database Server, а **nettype** задает используемый протокол. При использовании этих параметров для создания ярлыка настройки подключения будут сохранены, и пользователю потребуется помнить лишь собственный пароль.

Теперь щелкните ярлык дважды, чтобы протестировать соединение. Ошибки чаще всего возникают из-за того, что клиент не может установить соединение с сервером. В зависимости от используемого протокола существует несколько способов идентификации проблемы. (См. раздел Устранение Проблем при Установке Соединения с Базой Данных)

Если соединение было успешно установлено, просто повторите процедуру установки для каждого клиентского компьютера.

Выполнение в Качестве Сервиса

Если вы хотите запускать Dynamics NAV Database Server в качестве службы или использовали при запуске из командной строки параметр **installasservice** и используете протокол TCPS, вам следует убедиться, что служба запускается под учетной записью NT Authority\Network Service или Local System. Учетная запись NT Authority\Network Service существует только в Windows XP и Windows Server 2003. Если вы используете Windows 2000 Server, вам следует создать учетную запись для службы с наименьшим уровнем привилегий или использовать Local System.

Эта учетная запись должна иметь как минимум такие же привилегии как обычная пользовательская учетная запись и доменная учетная запись и не обладать привилегиями администратора домена или локального компьютера.

Чтобы обеспечить наивысший уровень безопасности, рекомендуется запускать Dynamics NAV Database Server на операционных системах Windows XP или Windows Server 2003, использовать протокол TCPS и запускать службу под учетной записью NT Authority\Network Service.

DRAFT

Следует не забыть добавить учетной записи NT Authority\Network Service или пользовательской учетной записи, под которой запущена служба права на доступ к файлу (файлам) базы данных, чтобы гарантировать, что пользователи смогут подключиться к базе данных.

Чтобы добавить права на чтение/запись файла базы данных учетной записи NT Authority\Network Service в Windows XP нужно:

1. В проводнике Windows найти папку, содержащую файл базы данных.
2. Выбрать файл базы данных, щелкнуть по нему правой кнопкой мыши и выбрать пункт Свойства.
3. В окне **Свойства**, щелкнуть вкладку **Безопасность**, затем щелкнуть кнопку **Добавить** (она расположена под полем **Группы и Пользователи**).
4. В окне Выбор: Пользователи, Компьютеры или Группы ввести **Network Service** и нажать OK.
5. **NETWORK SERVICE** будет добавлен в поле **Группы и Пользователи** окна **Свойства**.
6. Выберите **NETWORK SERVICE** и в поле **Разрешения** установите флажки **Чтение** и **Запись**.

Установка Опции Microsoft SQL Server для Использования с Dynamics NAV C/SIDE Клиент

Процедура установки при использовании опции SQL Server в корне отличается от описанной выше. Вначале нужно установить SQL Server на серверном компьютере в соответствии с документацией к Microsoft SQL Server.

Далее необходимо установить на серверном компьютере лишь один компонент с DVD-диска Dynamics NAV — две расширенные хранимые процедуры (extended stored procedures).

На серверном компьютере запустите установочный DVD -диск Dynamics NAV. В папке $:\SQL_ESP ($ обозначает букву, соответствующую приводу DVD) щелкните файл xp_ndo.exe. Этот файл содержит расширенные хранимые процедуры. Укажите программе путь к вложенной папке BINN, находящейся в той папке, где установлено ПО SQL Server. Распакованный файл xp_ndo.dll должен быть помещен в эту папку на сервере.

После того, как вы установили SQL Server и убедились, что он нормально функционирует, можно приступать к установке Dynamics NAV на клиентских компьютерах.

Завершающим шагом после установки клиентского ПО является создание на рабочем столе пользователя ярлыка и его тестирование. Ярлык должен содержать определенные параметры, например:

"C:\Program Files\...\finsql.exe" servername=MyServer,nettype=Named Pipes,ntauthentication=yes

Обратите внимание, что вместо исполняемого файла (при создании ярлыка) **fin.exe** нужно использовать **finsql.exe**.

Когда клиент Dynamics NAV устанавливает соединение с сервером в первый раз, программа просит загрузить лицензию на сервер.

Добавление Расширенных Хранимых Процедур Вручную

Используйте средства корпорации Microsoft, например, Enterprise Manager или SQL Server Management Studio, чтобы добавить файл xp_ndo.dll к расширенным хранимым процедурам, уже установленным в составе SQL Server. Названия расширенных хранимых процедур **xp_ndo_enumusergroups** и **xp_ndo_enumusersids**.

> **Примечание**
>
> Если вы хотите изменить модель безопасности, используемую в базе данных NAV SQL, вам следует загрузить обе расширенные хранимые процедуры на SQL Server.

Чтобы добавить расширенные хранимые процедуры:

DRAFT

1. Открыть Microsoft SQL Server Management Studio.

2. Раскрыть Server, раскрыть Databases, System Databases.

3. Раскрыть master, Programmability, Extended Stored Procedures.

4. Щелкнуть правой кнопкой мыши по Extended Stored Procedures, затем New Extended Stored Procedure, откроется окно **Extended Stored Procedure - New**.

5. Введите имя расширенной хранимой процедуру и выберите путь к библиотеке, где она хранится.

6. Щелкните Ok, чтобы добавить расширенную хранимую процедуру.

После того как расширенная хранимая процедура была успешно добавлена на сервер, ей должна быть назначена роль "public".

Чтобы назначить процедуре роль "public":

1. Открыть Microsoft SQL Server Management Studio.

2. Раскрыть Server, раскрыть Databases, System Databases.

3. Раскрыть master, Programmability, Extended Stored Procedures.

4. Щелкнуть правой кнопкой мыши по расширенной хранимой процедуре xp_ndo_enumusersids, затем **Properties**

5. Выберите раздел Permissions и установить разрешение на выполнение (Grant Execute) для роли "public" и пользователя "guest."

Повторите указанную процедуру для других расширенных хранимых процедур.

Установка Нескольких Стандартных Серверов

Возможно, перед вами встанет задача установить несколько стандартных серверов на одном компьютере. Помните о том, что сервисы, работающие на серверном компьютере, обычно используют протокол TCP/IP. Способ установки нескольких сервисов на одном компьютере основан на том, что два сервиса могут взаимодействовать на разном уровне, используя один и тот же протокол.

При использовании протокола TCP/IP необходимо изменить номера портов для сервисов. Чтобы изменить номер порта, нужно отредактировать файл **services** на серверном компьютере. Обычно этот файл находится в следующем каталоге Windows XP:

C:\WINDOWS\system32\drivers\etc\services.

Если вы не знаете, как редактировать этот файл, обратитесь за помощью к специалисту, знакомому с TCP/IP. После того, как необходимые изменения были произведены на сервере, следует сделать это для каждого клиента. Также вы можете изменить ярлык на каждом клиентском компьютере, чтобы задать имя сервера.

> **Примечание**
>
> Учтите, что при использовании более одного стандартного сервера могут существовать определенные ограничения. Один файл с лицензией предназначен для использования только с одним стандартным сервером

Устранение Проблем при Установке Соединения с Базой Данных

Проблема может возникнуть, если сервер C/SIDE Database Server функционирует нормально, но клиентские компьютеры не могут установить соединение с ним.

DRAFT

TCP/IP

Чтобы определить, почему клиенты Dynamics NAV не могут установить связь со стандартным сервером по протоколу TCP/IP, необходимо выполнить следующие действия:

1. Запустите клиентское ПО Dynamics NAV и попытайтесь установить соединение со стандартным сервером, используя имя сервиса (без префикса «C/SIDE Database Server»). Если это удалось, клиент может установить соединение с сервером и сервисом по протоколу TCP/IP, используя имя сервиса. Если это не удалось, Dynamics NAV не может использовать TCP/IP. Возможно, проблема связана с файлом hosts. Переходите к следующему пункту.
2. Используйте утилиту ping для проверки доступности сервера по IP-адресу (в командной строке введите: ping ###.###.###.###).
   Если это удалось, данный клиентский компьютер может установить соединение с серверным компьютером. Переходите к пункту 3. Если это не удалось, проблема связана с установкой соединения с серверным компьютером по TCP/IP, а не с Dynamics NAV. Перед тем, как продолжить, необходимо устранить эту проблему.
3. Используйте утилиту ping для проверки доступности сервиса по имени (без префикса " Dynamics NAV Server "; в командной строке введите: ping servicename). Учтите, что на самом деле вы не проверяете доступ к сервису. Вы, по сути, проверяете другим способом доступ к серверу.
   Если это удалось, файл hosts сконфигурирован корректно.
   Если это не удалось, значит в файле hosts отсутствует запись, позволяющая TCP/IP определить, где работает сервис. Добавьте в файл hosts строку, содержащую имя сервиса (без префикса «C/SIDE Database Server») и IP-адрес сервера.

Пакеты Обновлений и Обновления Системы Безопасности

Установка не считается завершенной, пока не будут установлены последние пакеты обновлений и обновления системы безопасности. Вы должны посетить Microsoft Update и установить соответствующие обновления на все компьютеры, на которых установлен Dynamics NAV. Также рекомендуется активизировать Автоматические Обновления на каждом компьютере, чтобы компьютеры могли получать критические обновления автоматически

DRAFT

ГЛАВА 2 ВАРИАНТЫ СЕРВЕРОВ

В данной главе будет рассказано о главных различиях при использовании Клиента Dynamics NAV C/SIDE со стандартной базой данных (C/SIDE Database Server) и базой данных в формате SQL (SQL Server).

Данная глава состоит из следующих разделов:

- Различные варианты серверов

2.1 Различные Варианты Серверов

Как было указано ранее, Dynamics NAV может работать с двумя разными серверами: C/SIDE Database Server и Microsoft SQL Server.

В данной главе будет приведено краткое описание двух вариантов серверов, поддерживающих работу с Dynamics NAV. Будут приведены основные отличия между ними и объяснены основные преимущества SQL Server над C/SIDE Database Server.

С точки зрения клиента, оба сервера работают и выглядят одинаково. Однако между ними существует ряд отличий:

- Способ создания базы данных.
- Доступные способы выполнения резервного копирования.
- Возможность доступа к данным в базе данных с помощью приложений сторонних производителей.
- Безопасность.
- Способ работы технологии SIFT.
- Мониторинг производительности
- Масштабируемость
- Поддержка нескольких процессоров

Далее эти темы будут рассмотрены подробно.

Создание Базы Данных

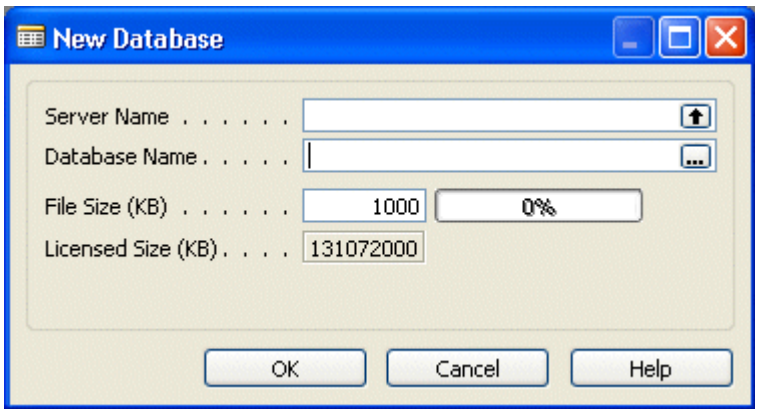
В зависимости от используемого варианта сервера, создание базы данных выполняется разными способами и при этом требуется принимать различные решения.

C/SIDE Database Server

Чтобы создать базу данных для C/SIDE Database Server, запустите клиента и щелкните меню File(Файл), Database(База данных), New(Создать), появится окно **Новая БД**. В данном окне можно:

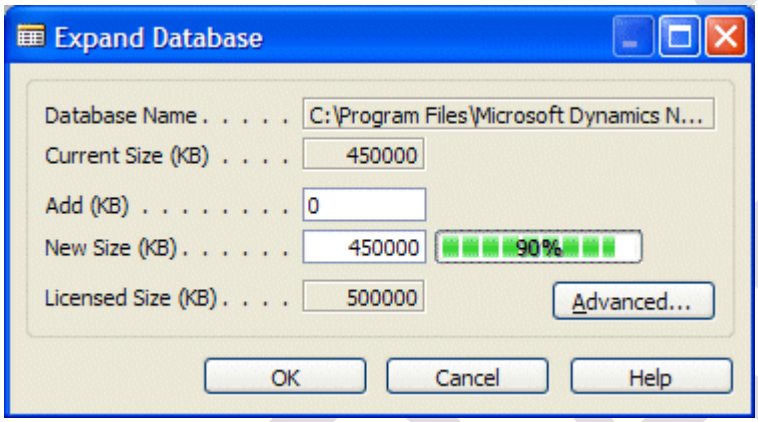
- Указать компьютер, на котором установлен сервер.
- Задать имя базы данных и ее расположение.
- Задать размер файла базы данных.

DRAFT

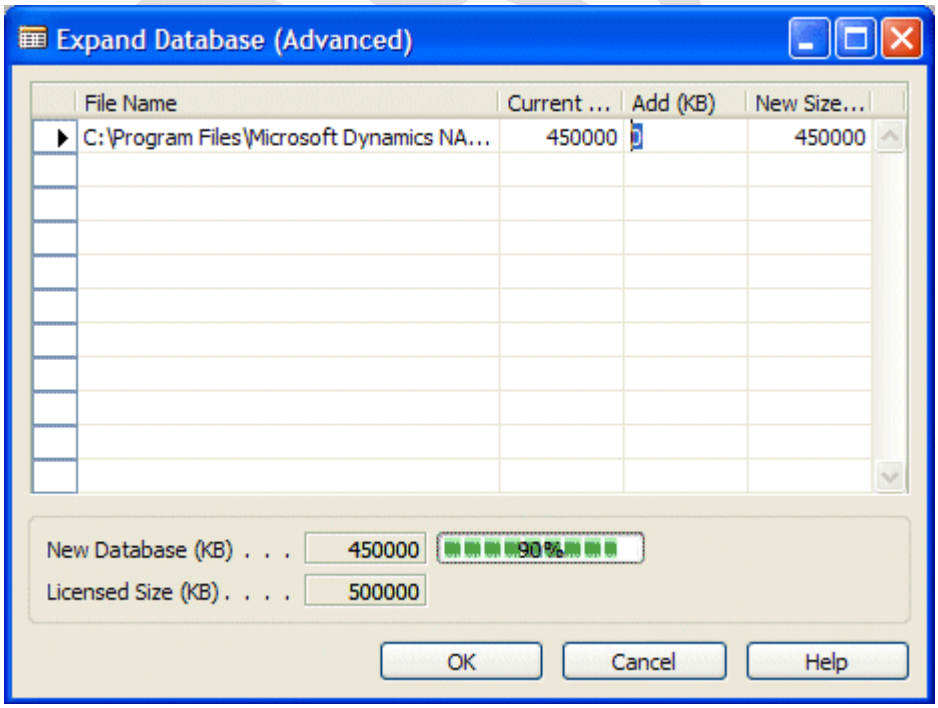


Если потребуется разделить базу данных на несколько файлов, чтобы хранить их на нескольких дисках, нужно будет расширить базу данных.

Чтобы расширить файл базы данных щелкните **File(Файл)**, **Database(База данных)**, **Expand(Расширить),** появится окно Расширить БД:



В данном окне можно задать прибавляемый объем к файлу базы данных. Если требуется, чтобы база данных состояла из нескольких файлов с различными расположениями, щелкните кнопку Дополнительно (Advanced), чтобы открыть окно Расширение БД (Дополнительно):



Каждая введенная строка в данном окне, представляет один файл базы данных. Рекомендуется, чтобы все файлы имели одинаковый размер, так как это обеспечит равномерное распределение нагрузки между различными файлами и дисками.

Большее количество физических дисков позволяет увеличить скорость и производительность C/SIDE Database Server. Однако база данных производительность C/SIDE Database Server может состоять максимум из 16 частей.

> **Важно**
>
> После создания БД, состоящей из нескольких файлов, следует завершить работу клиента Dynamics NAV и снова запустить его, чтобы сервер выполнил «Восстановление списка свободных блоков». Теперь можно восстановить базу данных Dynamics NAV из резервной копии. Теперь база данных равномерно распределена по всем частям базы данных и физическим дискам.

SQL Server Option

Перед тем как вы сможете подключиться к SQL Server 2005 из Microsoft Dynamics NAV, необходимо установить флаг трассировки 4616 в SQL Server 2005. При использовании SQL Server 2000 данный флаг устанавливать не требуется.

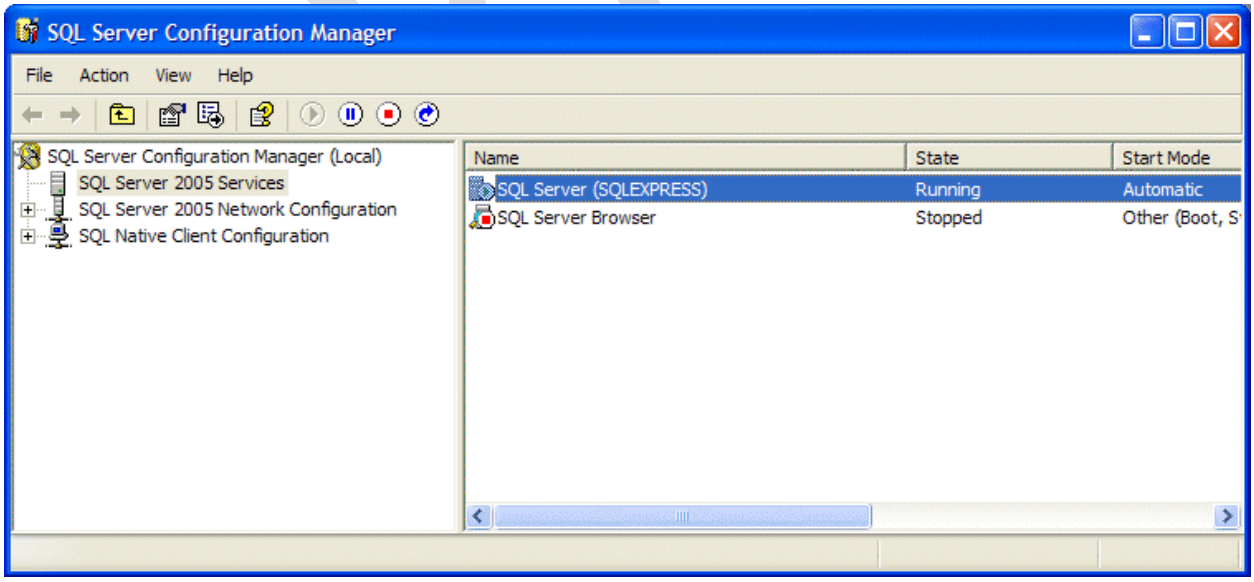
Если для установленного SQL Server задано имя, используемое по умолчанию (MSSQLSERVER), то флаг трассировки устанавливается автоматически. Ваша задача в этом случае не забыть перезапустить SQL Server. Если для SQL Server задано другое имя, то следует выполнить следующие шаги.

Включение флага трассировки

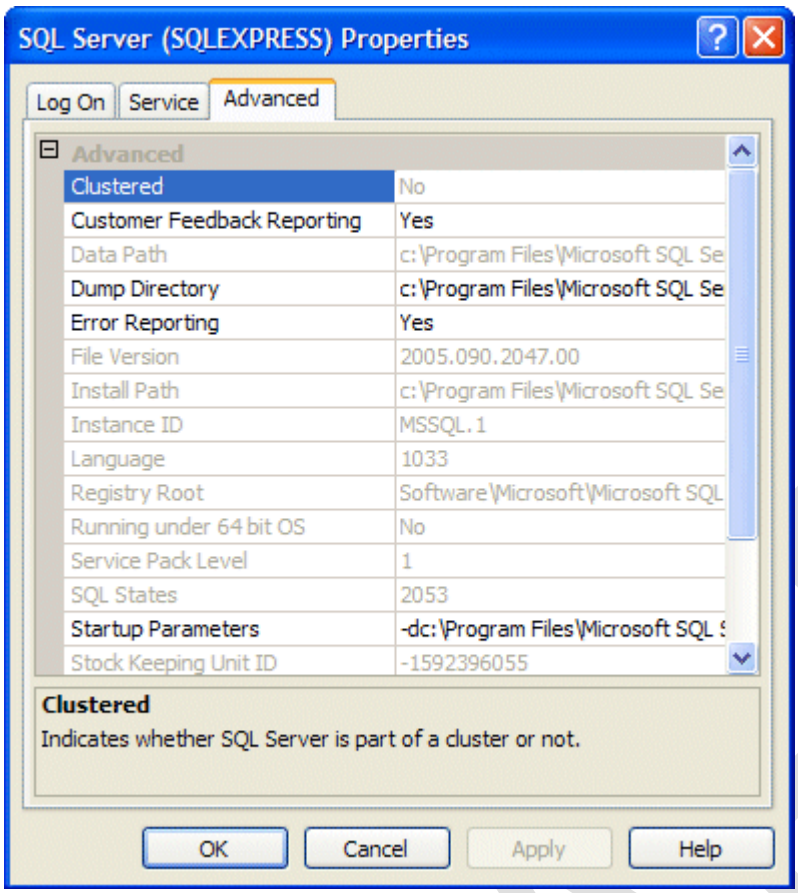
Для включения флага трассировки необходимо:

1. Открыть SQL Server Configuration Manager.

2. В левой панели, щелкнуть правой кнопкой мыши по SQL Server 2005 Services и выбрать пункт **Открыть**, чтобы увидеть все сервисы:

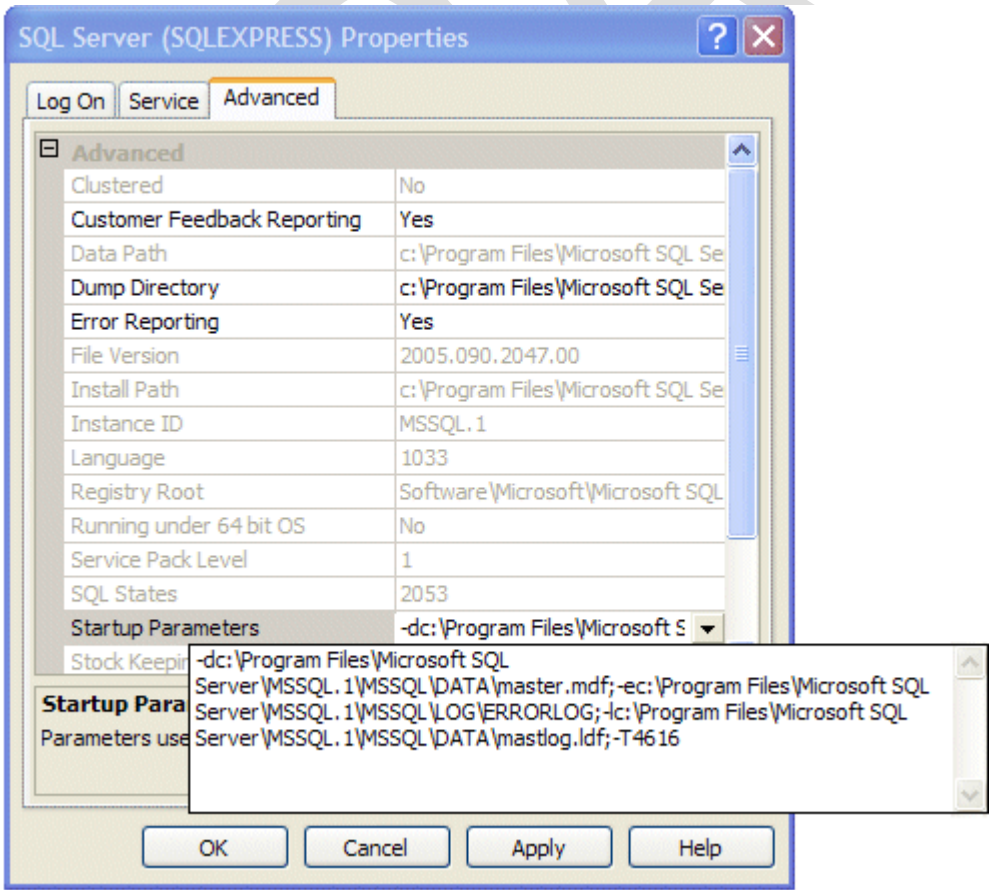


3. В правой панели, щелкнуть правой кнопкой мыши по SQL Server (MSSQLSERVER) или SQL Server (SQLEXPRESS) и выбрать пункт Свойства, чтобы открыть окно **Свойства (Properties)**:



4. В окне Свойства, открыть закладку ***Advanced*** и при необходимости раскрыть группу Advanced.

5. Щелкнуть по свойству Startup Parameters и раскрыть выпадающий список:



6. Ввести **;-T4616** в конце строки выпадающего списка.

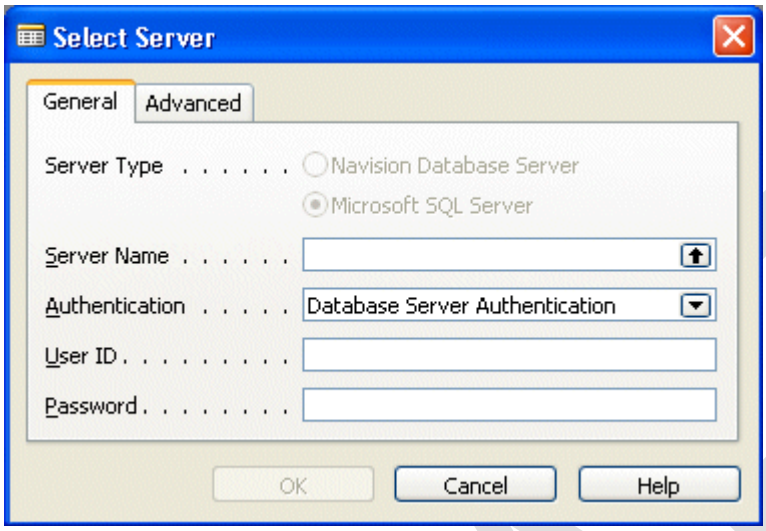
7. Перезапустить SQL Server.

Создание Базы Данных

Создание базы данных для SQL Server Option это последовательный процесс, при выполнении которого можно задать общие свойства для базы данных.

Для создания базы данных для SQL Server нужно выполнить следующие шаги:

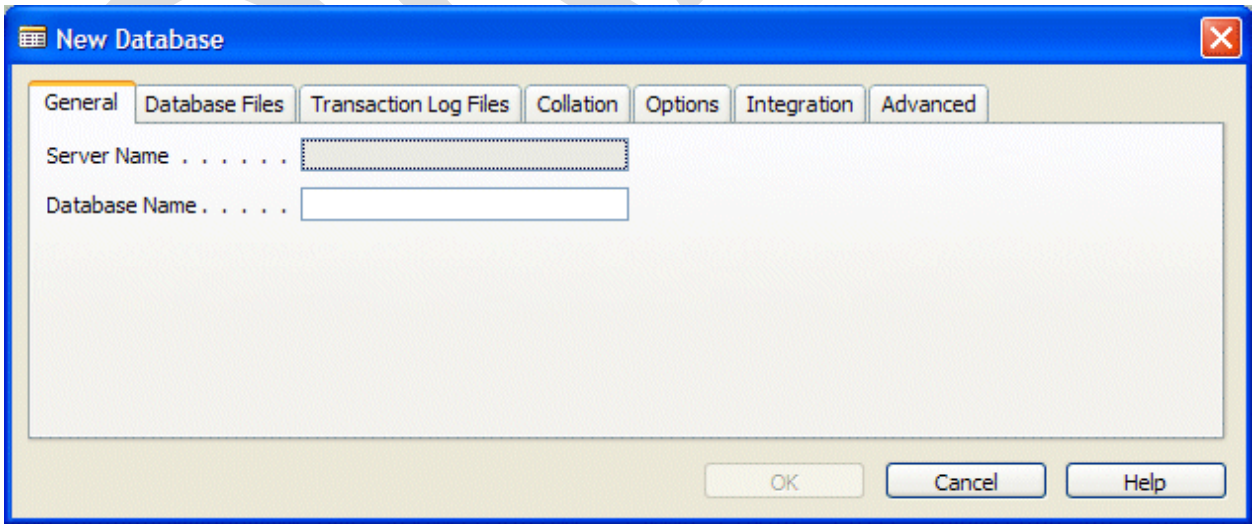
1. Щелкните меню File(Файл), Database(База данных), New(Создать), появится окно **Выберите Сервер**.



В данном окне можно:

- Указать сервер, на котором будет создана база данных.
- Указать тип аутентификации, который будет использоваться для входа на сервер.
- Если тип аутентификации =Аутентификация БД, ввести Код Пользователя и Пароль.
- Указать используемый Тип Сети (закладка Дополнительно).

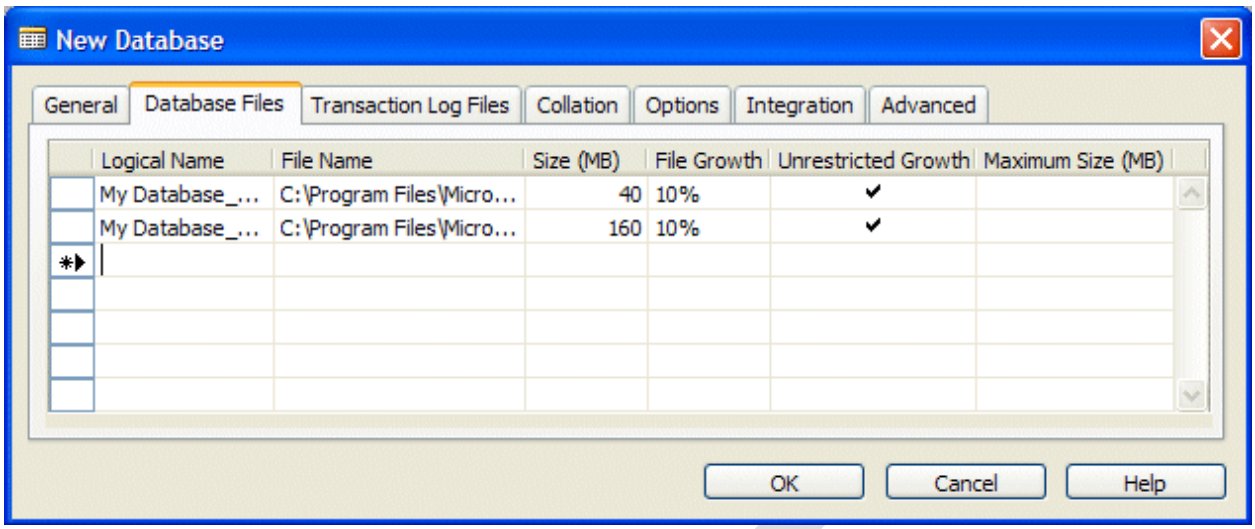
2. После выполнения указанных операций, щелкните кнопку OK, появится окно **Новая БД**:



Как можно видеть, окно содержит несколько закладок.

На закладке **Общее**, отображается имя выбранного сервера. В поле Имя БД нужно указать имя создаваемой базы данных.

DRAFT



На закладке **Файлы БД**, отображаются первые два созданных файла базы данных. Каждая введенная строка в данном окне представляет новый файл базы данных. Здесь можно задать размер файла и, где данный файл будет храниться. Более того, можно задать, как размер файла будет увеличиваться: будет ли увеличение выполняться без ограничений или для него задан максимальный размер.

На закладке **Файлы Журнала Транзакций**, можно выполнить такие же операции для файлов журнала транзакций, которые обновляются при использовании базы данных. Файл журнала транзакций содержит детальную информацию обо всех изменениях, выполняемых в базе данных. Эта информация используется при сбоях базы данных, например, для восстановления всех операций, которые были проведены с момента последнего резервного копирования.

На закладке **Сопоставление**, указывается используемый в базе данных тип сопоставления (Collation). Сопоставление наиболее важное свойство, задаваемое при создании базы данных. Сопоставление определяет способ упорядочивания и сортировки в базе данных и влияет на производительность базы данных. Изменение сопоставления после того как база данных создана является времяемким процессом.

SQL Server Option для Dynamics NAV позволяет выбрать один из двух вариантов: Сопоставление Windows или Сопоставление SQL

- Сопоставлениям Windows соответствуют сопоставлениям, поддерживаемым операционной системой Windows, где они настраиваются в Региональных и Языковых параметрах.
- Сопоставлениями SQL являются исходные сопоставления, введенные в SQL Server 7.0 и поддерживаемые для обратной совместимости.

При создании базы данных рекомендуется использовать сопоставление Windows. Данный тип сопоставления следует правилам сопоставления операционной системы.

На закладке **Параметры**, можно определить ряд параметров базы данных. При создании базы данных эти параметры содержат значения по умолчанию, которые подходят для большинства баз данных.

Эти параметры легко можно изменить после создания базы данных.

Здесь можно:

- Ограничить доступ к базе данных, выбрав режим Single User или, указав, что доступ к серверу имеют пользователи с определенными правами на SQL Server.
- Задать следующие параметры базы данных:

Поле	Описание

DRAFT

Поле	Описание
Recovery model	Эта настройка определяет то, какая информация регистрируется в журнале транзакций и, следовательно, какая схема восстановления будет использоваться для этой базы данных. Доступны следующие параметры: • Bulk-Logged • Full • Simple
Bulk-Logged	В случае выбора варианта Bulk-Logged, журнал транзакций будет содержать ограниченную информацию о некоторых операциях, в которых задействован значительный объем данных (например, о копировании больших массивов). Модель восстановления Bulk-Logged обеспечивает защиту от сбоев носителя информации, а также наиболее эффективное выполнение крупномасштабных операций с минимальным использованием пространства, отведенного под журнал. Модель восстановления bulk-logged состоит из следующих компонентов: • Резервное копирование БД. • Дифференциальное резервирование (необязательный компонент).
Full	В случае выбора варианта Full в журнале транзакций хранится подробная информация о каждой транзакции. Эта информация может использоваться при резервном копировании журнала транзакций. В модели полного восстановления используется резервное копирование базы данных и резервное копирование журнала транзакций, что позволяет обеспечить полную защиту от сбоев носителя информации. В случае повреждения одного или нескольких файлов данных процедура восстановления носителя информации позволяет восстановить все завершенные транзакции. Незавершенные транзакции при этом отменяются. Модель полного восстановления позволяет восстановить то состояние базы данных, которое было непосредственно перед сбоем, а также ее состояние на любой заданный момент времени. Чтобы обеспечить высокую способность к восстановлению, в журнал полностью записываются все операции, в том числе операции с большим объемом данных – такие как SELECT INTO, CREATE INDEX, а также операции по массированной загрузке данных. Модель восстановления Full (Полная) состоит из следующих компонентов: • Резервное копирование БД. • Дифференциальное резервирование (необязательный компонент). • Резервное копирование журнала транзакций.

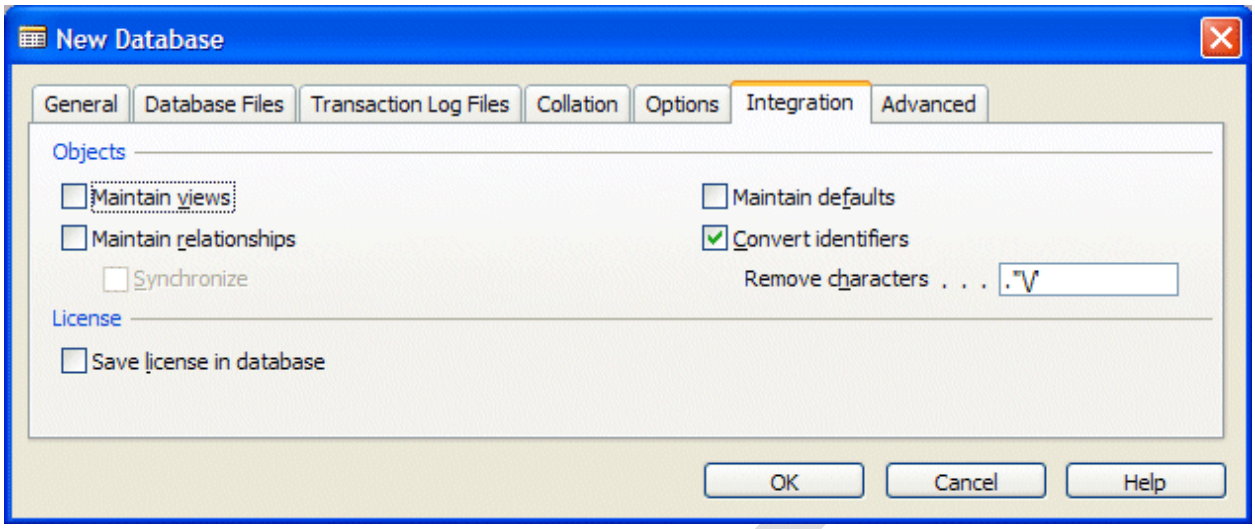
DRAFT

Поле	Описание
Simple	В случае выбора варианта Simple можно восстановить состояние базы данных на момент последнего создания резервной копии. Однако, восстановить ее состояние на момент сбоя или на другой заданный момент времени невозможно. Для этого следует использовать модель восстановления Full или Bulk-Logged. Модель восстановления Simple (Простая) состоит из следующих компонентов: • Резервное копирование БД. • Дифференциальное резервирование (необязательный компонент).
ANSI NULL default	С помощью этой настройки можно указать, будет ли к описаниям столбцов и к пользовательским типам данных по умолчанию применяться параметр NULL. В случае выбора этого варианта, для всех пользовательских типов данных и для всех столбцов, в описании которых не указан явно параметр NOT NULL, разрешаются записи типа NULL. Столбцы, в описании которых предусмотрены ограничения, подчиняются правилам этих ограничений; в этом случае данная настройка не действует.
Recursive triggers	Триггеры могут функционировать с прямой или косвенной рекурсией. Прямая рекурсия имеет место, когда триггер запускает выполнение операции, которая в свою очередь включает этот же триггер. Косвенная рекурсия имеет место, когда триггер запускает выполнение операции, которая включает триггер в другой таблице. Этот триггер изменяет данные в первой таблице, в результате чего снова включается первый триггер.
Torn page detection	В случае выбора этого варианта, операции ввода/вывода, незавершенные из-за сбоя питания или иного нарушения работы системы, будут отслеживаться SQL-сервером. Обычно такие операции обнаруживаются при повторном запуске SQL-сервера после сбоя. В этом случае необходимо восстановить базу данных из резервной копии, используя все имеющиеся резервные копии журнала транзакций.
Auto shrink	Этот параметр позволяет указать, будет ли выполняться периодическое автоматическое уплотнение файлов данных и файлов журнала транзакций SQL-сервером.
Разрешить Автопоиск при вводе	Данная настройка определяет, сможете ли вы использовать параметр Автопоиск при вводе при работе с функцией Поиск для нахождения операции в таблице или форме. Использование средства Автопоиск при вводе может повлиять на производительность, так как запросы на сервер отправляются для каждого введенного символа.

Данные параметры могут быть изменены позже.

На закладке **Интеграция**, содержатся настройки базы данных, влияющие на способ интеграции Microsoft Dynamics NAV с Microsoft SQL Server и внешними программными средствами. Эти настройки можно изменить позднее

DRAFT



Закладка интеграция разделена на две секции и содержит следующие параметры:

Поле	Описание
Поддерживать виды	Этот параметр определяет, будут ли на SQL-сервере создаваться и поддерживаться представления для каждого кода языка, добавляемого в поле или в таблицу Microsoft Dynamics NAV. Если эта опция активирована, SQL-представления будут использоваться внешними программными средствами для получения доступа к свойству объекта CaptionML на соответствующих языках (а не к имени, содержащемся в таблице).
Поддерживать связи	Этот параметр определяет, будут ли на SQL-сервере создаваться и поддерживаться ограничения внешних ключей для каждого свойства *TableRelation* таблицы Microsoft Dynamics NAV. Если этот параметр активирован, связи между таблицами Microsoft Dynamics NAV (ограничения внешних ключей) будут доступны внешним программным средствам. Эти связи блокированы и не используются для обеспечения целостности данных, но предназначены только для целей моделирования. Данное руководство содержит дополнительную информацию о связях таблиц в SQL Server
Синхронизировать	Этот параметр зависит от параметра Поддерживать Связи и активируется только при условии, что ранее была активирована поддержка взаимосвязей между таблицами на SQL-сервере. Данное руководство содержит дополнительную информацию о связях таблиц в SQL Server
Обработка умолчаний	Эта настройка определяет, будет ли SQL Server создавать и обслуживать стандартные ограничения для каждого поля в таблице Microsoft Dynamics NAV. При выборе данного параметра внешние программные средства смогут использовать стандартные значения при вставке или изменении данных в таблицах Microsoft Dynamics NAV.

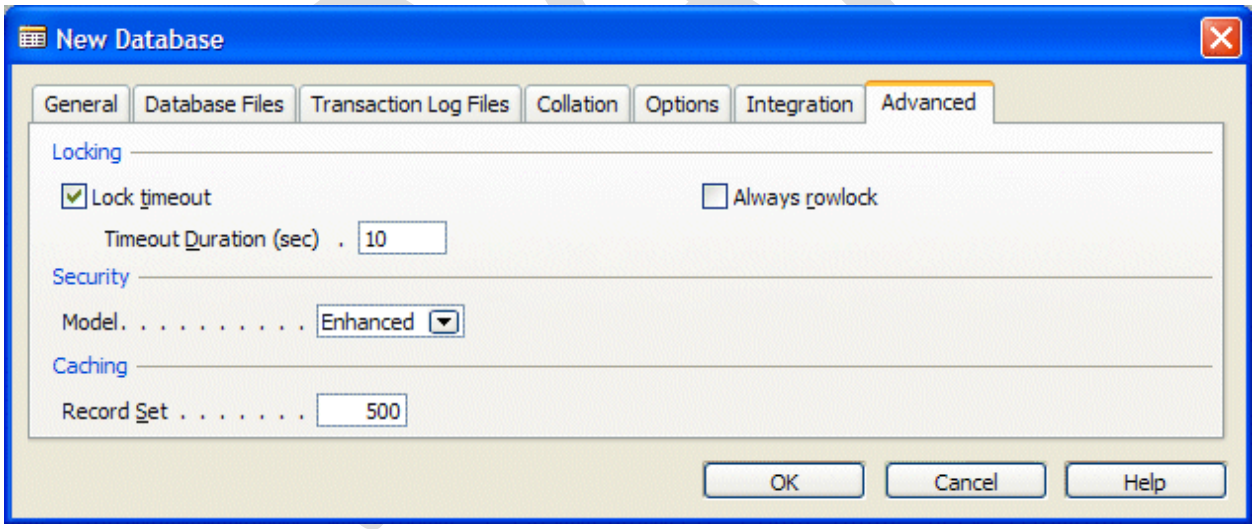
DRAFT

Поле	Описание
Конвертировать идентификаторы	Активизация этой опции позволяет выделять недопустимые символы в именах всех объектов Microsoft SQL Server (таблиц, столбцов, ограничений) и преобразовывать их в символ подчеркивания. В поле *Удалять символы* содержится список символов, подлежащих преобразованию в символ подчеркивания. Содержимое этого списка можно изменить. После выполнения преобразования необходимо закрыть и снова открыть базу данных, чтобы можно было использовать новые идентификаторы. Дополнительную информацию об идентификаторах и SQL Server, см. в разделе 6.4 Идентификаторы, Типы Данных, Форматы Данных в SQL Server для Dynamics NAV
Сохранять лицензию в базе данных	Если эта опция активирована, файл лицензии будет загружаться в базу данных и храниться в ней, а не на сервере. Это удобно в том случае, если один и тот же сервер используется несколькими базами данных, каждая из которых имеет отдельный файл лицензии. При выборе этого параметра во время создания или изменения базы данных вам будет предложено загрузить в базу данных файл лицензии.

Файлы базы данных для SQL Server не обязательно должны быть одинакового размера. После создания файлов базы данных, можно восстановить резервную копию базы данных Dynamics NAV.

Закладка **Дополнительно** содержит ряд параметров, позволяющих управлять способом обработки блокировки в базе данных.

На закладке содержатся следующие параметры



Поле	Описание
Ограничение таймаута	этот параметр позволяет указать, будет ли сеанс ожидать установки блокировки для ресурса, уже блокированного другим сеансом

Поле	Описание
Длительность тайм-аута (сек)	этот параметр позволяет указать максимальную продолжительность времени, в течение которого сеанс будет ожидать установки блокировки для ресурса, уже блокированного другим сеансом. Значение по умолчанию равно 10 секундам. Это значение можно изменить. Если этот флажок останется неустановленным, сеанс будет ожидать неограниченно долго.
Блокировка строк — всегда	этот параметр позволяет указать, чтобы в Microsoft Dynamics NAV всегда использовалась блокировка на уровне строк, а не блокировка на уровне страниц и таблиц.
Модель	Этот параметр позволяет указать, использует ли база данных Обычную или Расширенную модель безопасности. Значением по умолчанию является Расширенная.
Набор записей	Этот параметр позволяет указать количество записей, которые будут кэшироваться при выполнении в Microsoft Dynamics NAV операции FINDSET с параметром ForUpdate, равным FALSE. Если количество записей, считываемых оператором FINDSET, превышает указанное здесь значение, дополнительные операторы SQL будут переданы серверу, а производительность системы ухудшится. Увеличение этого значения увеличивает объем памяти, используемой клиентом для поддержки каждого оператора FINDSET. Значением по умолчанию является 500

Создание Резервных Копий

Два варианта серверов предоставляют очень разные возможности по созданию резервных копий и восстановления.

C/SIDE Database Server

При использовании C/SIDE Database Server, доступно два вариант создания резервных копий: создание резервной копии с помощью клиента и создание резервной копии на сервере. Инициация создания резервной копии с помощью клиента Dynamics NAV, выполняется из меню Сервис, Создание Копии БД.

Преимущество выполнения резервного копирования с помощью Dynamics NAV:

- База данных проверяется на наличие ошибок, так что в резервную копию не попадает ненужная информация.
- Данные упаковываются таким образом, чтобы они занимали как можно меньше места.
- Автоматически вычисляется объем пространства, которое необходимо выделить для резервной копии.
- При создании резервной копии с Dynamics NAV можно продолжать работать.

HotCopy

C/SIDE Database Server комплектуется программой создания резервных копий - HotCopy. Программа HotCopy выполняет резервное копирование быстрее, чем клиент. Она устанавливается вместе C/SIDE Database Server и хранится в том же каталоге, что и C/SIDE Database Server.

DRAFT

Программу HotCopy можно запускать только на сервере, она может сохранять резервные копии на жестком диске. Нельзя выполнять инкрементное или дифференциальное резервное копирование. Можно выполнить резервное копирование базы данных, в то время как она используется клиентом. Файлы, содержащие резервные копии не сжимаются.

SQL Server Option

Microsoft SQL Server поддерживает четыре различных типа создания резервных копий. Следует выбрать такой тип создания резервных копий, которые отвечает выбранному уровню безопасности.

Доступны следующие четыре типа создания резервных копий:

- Database backup – создание резервной копий всей базы данных.
- Transaction log backup – создание резервной копии всего журнала транзакций.
- Differential backup – выполняется создание резервной копии всех завершенных транзакций, выполненных с момента последнего создания резервной копии базы данных.
- File and filegroup backup – создаются резервные копии определенных файлов и групп файлов одной базы данных.

Указанные выше типы резервного копирования могут комбинироваться, что позволяет разработать свою собственную стратегию создания резервных копий и восстановления баз данных, соответствующую конкретной базе данных.

Для дополнительной информации о выполнении резервного копирования средствами SQL Server, обратитесь в документации Microsoft SQL Server.

Также можно выполнить резервное копирование/восстановление базы данных из клиента Dynamics NAV, работающего с SQL Server Option. Однако процедуры создания резервной копии/восстановление базы данных выполняемое средствами SQL Server выполняются непосредственно на сервере и поэтому значительно быстрее резервного копирования, выполняемого из клиента Dynamics NAV.

Восстановление резервной копии Dynamics NAV возможно непосредственно на SQL Server, без использования Dynamics NAV. Также возможно создать базу данных прямо в SQL Server, а затем в созданную базу данных можно восстановить из резервной копии базу данных Dynamics NAV.

SQL Server позволяет выполнять резервное копирование во время использования системы. В SQL Server можно автоматизировать многие задачи, выполняемые администратором, в том числе создание резервных копий. SQL Server позволяет устанавливать для базы данных план поддержки (с помощью Wizard), который включает оптимизацию базы данных, тестирование целостности и план создания резервных копий.

Одним из преимуществ SQL Server над C/SIDE Database Server – это возможность вести журнал транзакций. Журналы транзакций дают SQL Server возможность восстановить операции, выполненные вплоть до момента сбоя. Восстановление операций возможно, при восстановлении последней резервной копии и применения всех последующих журналов транзакций, для восстановления данных транзакций.

SIFT

Технология суммирования индексных полей (SumIndexField Technology) была разработана для улучшения производительности, при выполнении таких операций как расчет баланса клиента. В традиционных системах управления базами данных, такая операция приведет к серии запросов и вычислений, до того как будет получен результат.

DRAFT

C/SIDE Database Server

Мощь и эффективность SIFT на C/SIDE Database Server делает вычисление сумм для числовых столбцов в таблицах невероятно быстрым, даже в том случае, если таблица состоит из тысяч записей. Данная функциональность используется в Dynamics NAV повсеместно.

Представим, что пользователю требуется получить итоговое значение по полю Сумма. В традиционных системах управления базами данных (СУБД) эта операция выполняется получением каждой записи, и суммирования значений по полю Сумма, это очень длительная процедура в базах данных с тысячами записей. В Dynamics NAV, требуется создать вычисляемое поле (FlowField), указать формулу расчета этого вычисляемого поля на основании поля Сумма, после этого СУБД должна будет получить только одно значение из SumIndexField.

Технология SIFT основана на используемых в C/SIDE Database Server индексных полях (SumIndexFields), которые добавляются к индексам и требуют места в базе данных. Благодаря специальной структуре индексов, система затрачивает незначительное время на актуализацию сумм в SumIndexFields.

SQL Server Option

Способ реализации SIFT в SQL Server значительно отличается от реализации в C/SIDE Database Server. Дополнительную информацию о работе SIFT см. в разделе 6.6 Специальные Поля Таблиц.

Мониторинг Производительности.

Еще одним отличием между двумя вариантами сервером, являются способы мониторинга производительности.

Наиболее важным инструментом, используемым для мониторинга производительности приложения, является Монитор Клиента. Этот инструмент может использоваться с обоими вариантами серверов. При использовании Монитора Клиента в SQL Server Option, у него появляются некоторые опции и поля, благодаря которым, можно получить дополнительную информацию о ходе выполнения операций приложением.

Монитор Клиента – это важный инструмент, используемый для решения проблем с производительностью и блокировками. Также его можно использовать для определения худших обращений к серверу, а также проблем с индексами и фильтрами при использовании SQL Server Option. Монитор Клиента и Выполнение Кода теперь работают в тесном взаимодействии, что позволяет легко найти код, который генерирует определенные обращения к серверу.

Dynamics NAV также включает отладчик, который используется для улучшения функций, создаваемых на языке C/AL. Отладчик, также используется для обоих вариантов серверов.

При использовании SQL Server Option, указанные функции могут дополняться логом ошибок SQL Server (SQL Server Error Log). Если включить флаги трассировки 1204 и 3605, система будет генерировать дополнительные диагностические сообщения и записывать их в лог. Это даст дополнительную информацию о типах блокировок, которые приводят к взаимным блокировкам.

Детальное описание, как использовать данный инструментарий, а также общую информацию о решении проблем с производительностью см. в руководстве *Performance Troubleshooting Guide,* которое доступно на Dynamics NAV Tools CD. Tools CD содержит ряд дополнительных инструментов, которые также используются для решения проблем с производительностью.

Прочие Отличия

Прочие отличия между двумя вариантами серверов:

- Масштабируемость

Еще одним главным отличием между двумя вариантами серверов является масштабируемость.

DRAFT

SQL Server Option может поддерживать больше одновременно работающих пользователей, чем C/SIDE Database Server.

- Обеспечение Безопасность

SQL Server Option поддерживает систему безопасность на уровне записи (см. ГЛАВА 29 ПОДДЕРЖКА СИСТЕМЫ БЕЗОПАСНОСТИ НА УРОВНЕ ЗАПИСИ).

- Поддержка нескольких процессоров

C/SIDE Database Server, в отличие от SQL Server, не может использовать несколько процессоров.

- Доступ к базе данных с помощью приложений сторонних производителей

При использовании SQL Server Option для Dynamics NAV, доступ к данным с помощью приложений сторонних производителей упрощается.

DRAFT

ГЛАВА 3 ОБЕСПЕЧЕНИЕ БЕЗОПАСНОСТИ В СИСТЕМЕ DYNAMICS NAV

В данной главе вы познакомитесь с системой обеспечения безопасности Dynamics NAV. Будет рассказано о различных уровнях обеспечения безопасности, существующих в Dynamics NAV и принципах их работы.

Также будет приведено краткое описание функциональных и предметных областей Dynamics NAV. Будет разъяснена система лицензирования Dynamics NAV.

Данная глава состоит из следующих разделов:

- Система безопасности
- Функциональные области и Гранулы

3.1 Система Безопасности

Система безопасности Dynamics NAV позволяет управлять доступом к объектам (таблицам и т.п.) индивидуально для каждого пользователя, имеющего доступ к базе данных. Так же можно задать тип доступа к таблицам и записям, доступным пользователю, т.е. можно указать может ли он читать, изменять или добавлять данные.

Более того, при использовании SQL Server, есть возможность разрешить пользователю доступ к определенным записям, хранящимся в базе данных. Другими словами права доступа могут назначаться как на уровне таблиц, так и на уровне записей при условии, что используется SQL Server для Dynamics NAV.

Система безопасности Dynamics NAV содержит информацию о правах доступа определенных для каждого пользователя, имеющего доступ к базе данных, отдельно для каждой базы данных.

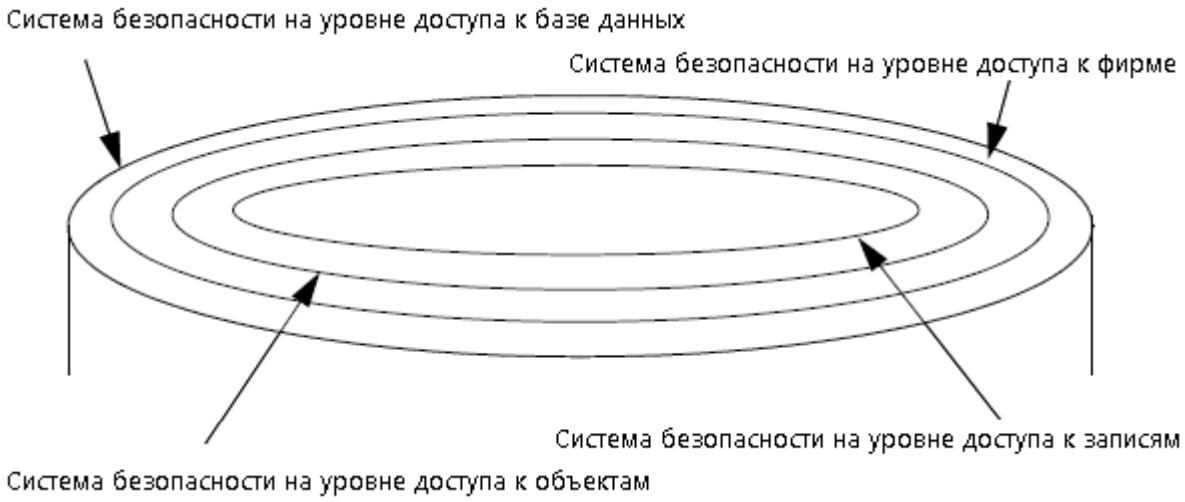
Эта информация включает роли, назначенные пользователю, и набор прав доступа, заданных для каждой роли.

Система безопасности Dynamics NAV – это однородная, интегрированная система, состоящая из четырех уровней безопасности:

- На уровне доступа к базе данных (Database Level Security)
- На уровне доступа к фирме (Company Level Security)
- На уровне доступа к объектам (Object Level Security)
- На уровне доступа к записям (Record Level Security)

Графически это можно представить, как луковицу, состоящую из нескольких слоев, где сердцевина – это запись в базе данных:

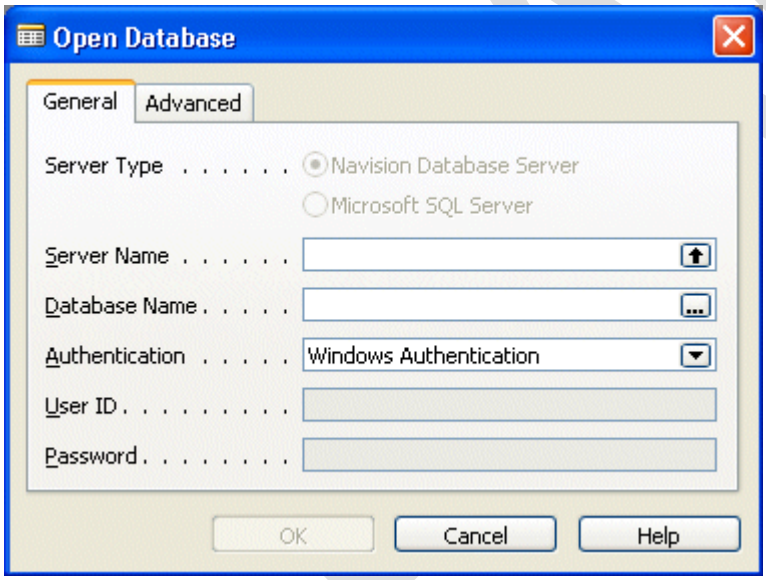
DRAFT



Система Безопасности на Уровне Доступа к Базе Данных

Первый уровень системы безопасности, с которым пользователь встречается при открытии базы данных это система безопасности на уровне доступа к базе данных. При запуске программы и попытке открыть базу данных, система проверяет полномочия пользователя и возвращает сообщение об ошибке, если полномочий недостаточно.

Чтобы открыть базу данных надо щелкнуть Файл, База Данных, Открыть, появится окно **Открыть БД**:



В данном окне, используя кнопку Выбор, надо указать сервер, к которому требуется получить доступ и открываемую базу данных.

В поле Аутентификация, указывается тип аутентификации, который будет использоваться при проверке полномочий на доступ к базе данных.

Dynamics NAV поддерживает два типа аутентификации:

- Аутентификация Windows
- Аутентификация БД

Тип аутентификации надо выбирать на основании типов регистрационных имен, заведенных в базе данных.

Регистрационные Имена Windows

Если в Dynamics NAV используется аутентификация Windows, для пользователей указываются регистрационные имена Windows. Регистрационные имена Windows соответствуют логинам пользователей Windows, а также группам в домене. Управление регистрационным именами Windows осуществляется в отдельном окне.

Если используется тип аутентификации Windows, при подключении к базе данных, пользователю не требуется вводить код пользователя и пароль. Dynamics NAV автоматически запрашивает Windows, существует ли у данного пользователя права на доступ к серверу. Если пользователю разрешено подключение к серверу, Dynamics NAV проверяет, сопоставлено ли с данным пользователем регистрационное имя Windows в Dynamics NAV. Если для пользователя существует регистрационное имя Windows, ему будет предоставлен доступ к базе данных. Пользователю будут присвоены права согласно регистрационному имени Windows для пользователя или для группы пользователей, в которую входит данный логин Windows.

Если пользователь не имеет соответствующего регистрационного имени Windows или для данного регистрационного имени Windows нет прав на вход в базу данных, аутентификация не будет выполнена, вместо этого пользователю будет показано сообщение об ошибке.

Регистрационные имена БД

При использовании регистрационных имен БД, пользователям должны быть присвоены Коды Пользователя и Пароли.

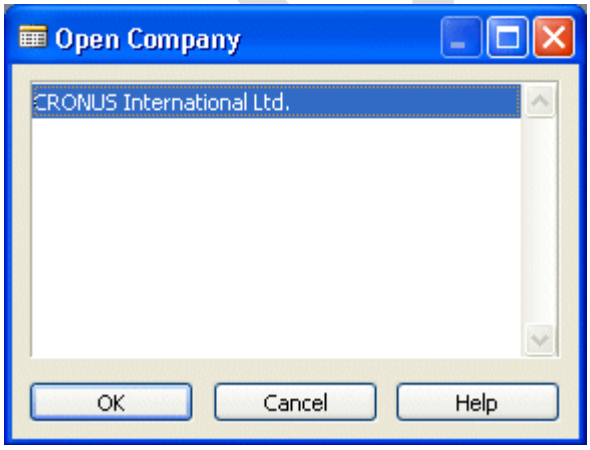
При использовании SQL Server пользователи также должны иметь логины на SQL Server. SQL Server выполняет собственную аутентификацию логинов и паролей пользователей. SQL Server проверяет, существуют ли логины и пароли на SQL Server, соответствующие Коду пользователя. Администратор SQL Server должен создать эти логины заранее, с помощью инструментария SQL Server. Если на SQL Server логины не созданные, аутентификация не будет выполнена, вместо этого пользователю будет показано сообщение об ошибке.

После выполнения аутентификации, пользователь получает доступ к серверу. Затем система безопасности базы данных проверяет права доступа пользователя на основании прав доступа заданных для регистрационного имени БД. В правах доступа заданных для регистрационного имени БД, указаны объекты базы данных, к которым пользователь имеет доступ, например, таблицы. Также в правах доступа содержатся дополнительные права, позволяющие пользователю самостоятельно изменять базу банных.

Системы Безопасности на Уровне Доступа к Фирме

После получения доступа к базе данных, пользователь может открыть фирму, с которой он собирается работать.

Чтобы открыть фирму щелкните Файл, Фирма, Открыть. Появится окно **Открыть Фирму**:



В данном окне представлен список всех фирм, созданных в текущей базе данных, к которым пользователю разрешен доступ. В базе данных Dynamics NAV может храниться несколько фирм.

DRAFT

Фирмы могут иметь таблицы, доступные только им, или же разделять доступ к некоторым таблицам с другими фирмами.

Выберите фирму, которую требуется открыть, и щелкните OK. Если в базе данных есть фирмы, для открытия которых у пользователя нет прав, то такие фирмы в окне не отображаются.

Система Безопасности на Уровне Доступа к Объектам

После того как фирма была открыта, возможности пользователя по работе с ней также контролируются системой безопасности Dynamics NAV.

Система безопасности Dynamics NAV состоит из ролей и прав доступа, которые присваиваются пользователям, имеющим доступ к фирме. Роли Dynamics NAV определяют доступ и действия, которые пользователь может выполнять над объектами в базе данных.

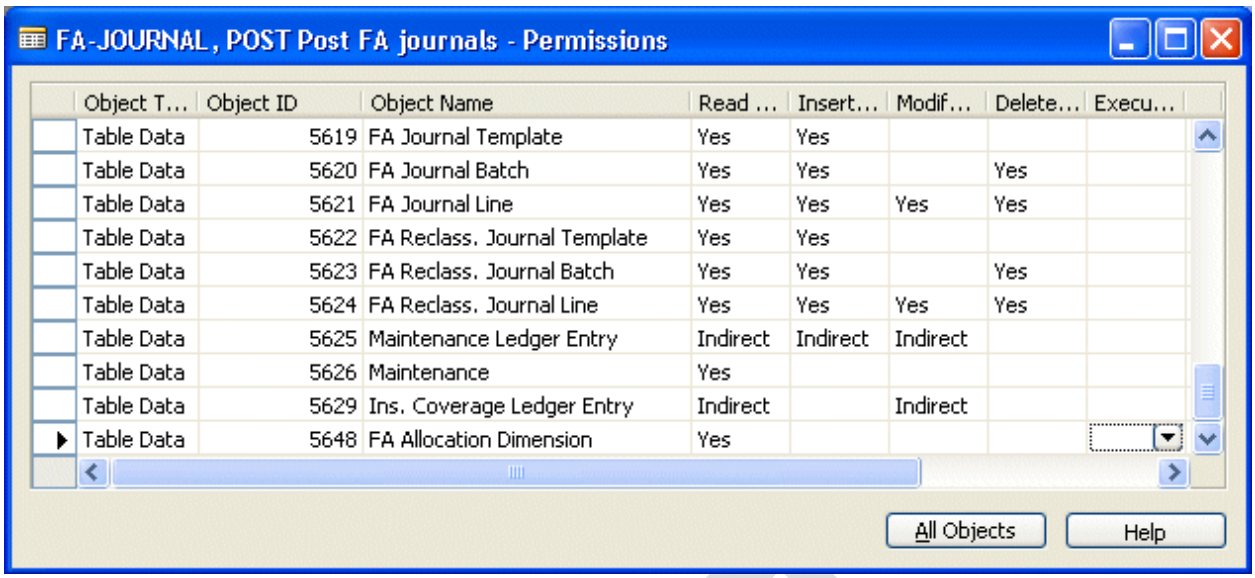
Система безопасности Dynamics NAV рассматривает базу данных как совокупность следующих объектов:

Объект	Описание
Данные (Table Data)	Данные хранящиеся в таблицах.
Таблица (Table)	Сами таблицы.
Форма (Form)	Формы, которые используются для просмотра и ввода данных.
Отчет (Report)	Отчеты, используемые для представления данных.
Датапорт (Dataport)	Датапорты используются для импорта и экспорта данных.
Функция (Codeunit)	Функции (также используется термин Программный модуль), используемые в базе данных.
XMLport (XMLport)	XMLport используются для импорта и экспорта данных в формате XML.
MenuSuite (MenuSuite)	Объект, который содержит меню, отображаемые в Панели Навигации.
Система (System)	Системные таблицы в базе данных, которые позволяют пользователям выполнять резервное копирование, изменять файлы лицензии и т.п.

Различные роли, существующие в базе данных, определяют действия, которые пользователь может выполнять над объектами.

На следующем рисунке показано как назначаются права доступа:

DRAFT



Object T...	Object ID	Object Name	Read ...	Insert...	Modif...	Delete...	Execu...
Table Data	5619	FA Journal Template	Yes	Yes			
Table Data	5620	FA Journal Batch	Yes	Yes		Yes	
Table Data	5621	FA Journal Line	Yes	Yes	Yes	Yes	
Table Data	5622	FA Reclass. Journal Template	Yes	Yes			
Table Data	5623	FA Reclass. Journal Batch	Yes	Yes		Yes	
Table Data	5624	FA Reclass. Journal Line	Yes	Yes	Yes	Yes	
Table Data	5625	Maintenance Ledger Entry	Indirect	Indirect	Indirect		
Table Data	5626	Maintenance	Yes				
Table Data	5629	Ins. Coverage Ledger Entry	Indirect		Indirect		
Table Data	5648	FA Allocation Dimension	Yes				

На данном рисунке показаны некоторые права доступа, которые были определены для роли ОС-ЖУРНАЛ, УЧЕТ (FA – Journal, Post). Как можно видеть, для роли определены права доступа на выполнения различных действий над большим списком объектов. Права доступа, которые могут быть заданы для объекта, бывают следующих типов:

Права доступа	Описание
Чтение (Read)	Пользователю предоставляется право на чтение указанного объекта
Вставка (Insert)	Пользователю предоставляется право на вставку данных в указанный объект.
Изменение (Modify)	Пользователю предоставляется право на изменение данных в указанный объект.
Удаление (Delete)	Пользователь может удалять данные из указанного объекта.
Запуск (Execute)	Пользователь может запускать указанный объект.

Если пользователю присвоены права на чтение формы, он может открыть ее и посмотреть отображаемые в ней данные. Однако если у него нет прав на вставку, то он не может вводить новые данные в эту форму.

Некоторые формы, при открытии отображают данные полученные из нескольких таблиц. Чтобы открыть данную форму, требует права на просмотр всех данных, отображаемых на форме. Пользователю не обязательно должны предоставляться права на чтение непосредственно данных из таблиц. Достаточно будет указать ему косвенные права на просмотр данных, отображаемых на форме. Косвенные права к таблице означают, что пользователь не сможет открыть таблицу и посмотреть в ней данные, однако он может получить доступ к данным косвенно, через другие объекты, например форму или отчет, к которым у него есть прямой доступ.

Dynamics NAV поставляется с предопределенным набором стандартных ролей. Вы можете использовать данные роли, как они поставляются, или изменить их под конкретные требования. Также можно создать свои собственные роли с нужными правами доступа.

Система Безопасности на Уровне Доступа к Записям

Система безопасности на уровне записи, позволяет ограничивать доступ пользователя к таблице, указав в правах доступа определенные записи, к которым он имеет доступ в данной таблице. Система безопасности на уровне записи доступна исключительно при использовании SQL Server для Dynamics NAV.

DRAFT

Система безопасности на уровне записи основана на применении фильтров защиты к таблицам и данным, к которым пользователь имеет доступ. Например, есть возможность указать, что пользователь может видеть записи, содержащие информацию об определенном клиенте, и не может видеть информацию по другим клиентам.

Для получения дополнительной информации об установке фильтров защиты см. руководство *Installation & System Management: SQL Server Option for the C/SIDE Client*.

Для дополнительного описания применения системы безопасности на уровне записи см. ГЛАВА 29 ПОДДЕРЖКА СИСТЕМЫ БЕЗОПАСНОСТИ НА УРОВНЕ ЗАПИСИ.

Что Необходимо Знать о Системе Безопасности Dynamics NAV

При использовании системы безопасности Dynamics NAV следует помнить следующее:

- Система безопасности Dynamics NAV активизируется при создании первого Регистрационного Имени. До этого времени, любой может выполнять в базе данных Dynamics NAV операции, какие пожелает. Кроме того, первое заведенное в систему регистрационное имя должно иметь привилегии superuser (SUPER). Пользователи с привилегиями superuser имеют доступ ко всей базе данных Dynamics.
- Пользовать может назначать права доступа другому пользователю, только в пределах тех прав, которыми он сам располагает. Поэтому рекомендуется, чтобы администрирование доступа осуществлял пользователь с правами superuser (SUPER).
- Одной из первых вещей которую должен сделать администратор, это создать регистрационный имена для других сотрудников, которые должны иметь доступ к базе данных и определить им права доступа.
- В таблице Dynamics NAV может присутствовать вычисляемое поле (FlowField) которое рассчитывает суммы на основании данных хранящихся в другой таблице. В этом случае пользователь должен иметь права доступа на чтение данных из обеих таблиц или он не сможет просматривать первую таблицу.

Дополнительное описание об управлении системой безопасности в Dynamics NAV, включая подробные инструкции по созданию регистрационных имен и ролей, а также о том, как назначать и изменять права доступа приведено в руководствах *Installation & System Management* для каждого варианта используемых серверов.

3.2 Функциональные Области и Гранулы

После того как мы познакомились с различными уровнями системы безопасности, которые доступны в Dynamics NAV, следует перейти к функциональным областям Dynamics NAV и узнать как файл лицензии контролирует доступ к функциональным областям, на основании содержащейся в нем информации.

Стандартное приложение Dynamics NAV состоит из нескольких предметных областей. Каждая предметная область в свою очередь состоит нескольких функциональных областей. Таким образом, функциональные области представляют большой набор различной функциональности, при этом каждому клиенту, возможно, потребуется лишь часть из этой функциональности. Для этого используется файл лицензии Dynamics NAV, в котором перечислены гранулы. Каждая гранула представляет определенную маленькую функциональную область.

Таким образом, обеспечивается возможность для клиента приобретать именно такую лицензию, которая позволит его сотрудникам иметь доступ только к той функциональности, которая им нужна.

Dynamics NAV содержит следующие предметные и функциональные области:

Предметная область	Функциональная область

DRAFT

Предметная область	Функциональная область
Финансы	Финансовые операции Денежные средства Клиенты Расчеты Поставщики Расчеты Основные средства Наличие
Продажи и Маркетинг	Продажи Обработка заказов Маркетинг Товары и цены Анализ и отчетность История
Покупка	Планирование Обработка заказов Товары и себестоимость Анализ и отчетность История
Склад	Заказы и Контракты Планирование и исполнение Обработка товаров Наличие История
Производство	Проектирование товаров Производственные мощности Планирование Исполнение Учет себестоимости История
Работы	Работы Отчеты История Периодические задания
Планирование ресурсов	Ресурсы Отчеты История Периодические задания

DRAFT

Предметная область	Функциональная область
Сервис	Контакты Планирование и отправка Обработка заказов История
Персонал	Сотрудники Простой Регистрация Отчеты
Администрирование	Управление ИТ Настройка приложения

Каждая из этих функциональных областей представляет важную область деятельности. Однако даже если клиент имеет доступ к Финансовым операция, он не обязан использовать всю доступную в этой области функциональность.

Далее приведена таблица, в которой в которой перечислены гранулы функциональной области Финансы:

Название и номер Гранулы	**Описание**
Финансы (3,010)	Гранула предназначена для создания фирмы и настройки ее учетной политики. В нее входят все функциональные средства, необходимые для этого: план счетов, финансовые журналы, функции обработки НДС, типовые журналы и коды аудита, внутренняя и внешняя отчетность. Также включены функции для формирования внутренней и внешней отчетности. Гранула позволяет осуществлять учет операций и генерировать отчетность в базовой валюте. В случае приобретения гранулы Мульти Валютность, возможен учет и формирование отчетов в дополнительных валютах. По умолчанию, данная гранула предусматривает работу с двумя языками - English US и родной язык вашей компании. Эта гранула позволяет 1 сессию NAV Application Server. Гранула в обязательном порядке должна присутствовать в любой конфигурации системы, так как она включает 1 сессию и первую фирму. Требования: Dynamics NAV Version 5.XX

DRAFT

Название и номер Гранулы	Описание
Бюджеты (3,030)	Гранула позволяет работать с бюджетами по финансовым счетам. После того, как бюджет сформирован, может быть напечатан баланс, отражающий процентную разницу между бюджетными значениями и фактическими показателями. Возможна одновременная работа с несколькими бюджетами. Как правило, бюджет формируется за определенный период времени по выбранным финансовым счетам. Как уже отмечалось, одновременно можно работать сразу с несколькими бюджетами: формировать, копировать данные из одного бюджета в другой. Например, может существовать 100% бюджет, 110% бюджет и т.д. Требования: Главная Книга
Финансовые отчеты (3,040)	Гранула позволяет генерировать финансовую отчетность. Данная отчетность может базироваться на показателях из плана счетов или бюджетов. При этом внешний вид и содержательное наполнение финансовых отчетов могут существенно отличаться от принципов структурирования и организации плана счетов. Гранула Финансовая Отчетность является фильтром плана счетов, позволяющим отбирать счета, которые вы хотите включить/исключить из отчетности. Она также используется для изменения порядка перечисления финансовых счетов, или объединения показателей самыми разнообразными способами. При помощи инструментария, реализованного в системе, возможно, изменить внешний вид отчетов и раскладку столбцов. Кроме этого, поддерживается возможность несложных калькуляций и расчетов. Требования: Главная Книга
Консолидация (3,050)	Гранула позволяет осуществлять консолидацию фирм в Dynamics NAV. Фирмы могут находиться в одной или нескольких базах данных Dynamics NAV или в другом файле. В грануле Консолидация предусмотрены специальные средства импорта/экспорта информации. Если используемые данные взяты из нескольких баз данных Dynamics NAV, гранула может быть использована только материнской компанией. Требования: Главная Книга
Распределения (3,020)	При помощи специальных ключей, гранула дает возможность распределять финансовые операции по счетам, отделам и проектам. Ключи распределения задаются, исходя из суммы, процента или количества. Функция распределения используется в самых разных ситуациях, например, для разноски накладных расходов (таких как рента) по центрам затрат компании. Требования: Главная Книга

DRAFT

Название и номер Гранулы	Описание
Центры ответственности (3,060)	В этой грануле можно настроить центры прибыли и/или центры затрат. Компания может продавать товары по специальным ценам в зависимости от центра ответственности. Эта функциональность предоставляет возможность связать пользователя с центром ответственности так, чтобы отражались только документы продажи и покупки, связанные с определенным пользователем. Кроме того, у пользователя упрощается ввод дополнительных данных, таких как измерения или коды складов. Требования: Мульти Склады
Базовый XBRL (3,070)	Гранула позволяет экспортировать документы из Dynamics NAV в формат XBRL и импортировать таксономии XBRL в Dynamics NAV из Интернета, электронной почты или прочих систем. XBRL – это спецификация на основе XML, которая использует принятые стандарты финансовых отчетов на основе стандартизированных основных тегов с данными. Можно преобразовать вашу главную книгу в таксономии XBRL, что означает, что один и тот же отдельных документ может быть использован для различных целей, независимо от требуемого получателем формата документа. Требования: Главная Книга
Журнал изменений (3,080)	Эта гранула позволяет вам протоколировать изменения пользователя, произведенные в мастер-данных Dynamics NAV. Можно протоколировать все прямые модификации, которые делает пользователь с данными в базе данных, за исключением изменений в «рабочих документах», таких как журналы и заказы покупки/продажи. Функциональность журнала изменений позволяет получить хронологический список всех измерений в любом поле в любой таблице (за исключением вышеуказанных) и увидеть кто (Код пользователя) произвел такие изменения. Требования: Главная Книга

Информация, приведенная в данной таблице, может не совпадать с составом и определением гранул поставляемых Microsoft Business Solution в настоящее время.

Таким же образом любая функциональная область может быть детализирована до конкретной функциональности.

Это означает, что лицензионный файл и то, каким образом он составлен, имеет большое влияние на то, какая функциональность будет доступна при запуске Dynamics NAV. Лицензионный файл и гранулы, указанные в нем определяют доступные функциональные области и возможные для использования функции.

Преимуществом такой системы является то, что клиент платит только за те гранулы, которые нужны для его бизнеса.

DRAFT

ЧАСТЬ 2 ОСНОВЫ

ГЛАВА 4 ОСНОВЫ СРЕДЫ C/SIDE

В среде C/SIDE имеется семь типов объектов приложения. Каждый тип прикладного объекта создан с использованием специального инструмента под названием «дизайнер». Функционирование всех прикладных объектов, создаваемых с использованием этих дизайнеров, основано на единых принципах. Глубокое знание этих принципов позволяет ускорить процесс разработки приложения C/SIDE. В настоящем разделе вы познакомитесь с интерфейсом C/SIDE и общими принципами, на которых основано функционирование прикладных объектов C/SIDE:

- Интерфейс C/SIDE;
- Что такое приложение C/SIDE;
- Физическая и логическая базы данных.

4.1 Интерфейс C/SIDE

В данном разделе вы познакомитесь с интерфейсом C/SIDE и некоторыми основными концепциями, связанными с созданием приложений, такими как разные типы объектов.

Создание Объектов Приложения

Среда C/SIDE основывается на семи типах объектов приложения:

Таблица (Table). Таблица используется для хранения данных. Обычно бизнес-приложение содержит таблицу Customer, в которой хранится информация о каждом клиенте, такая как имя, адрес, телефонный номер и контактное лицо. Понимание таблиц – это ключ к использованию всех остальных объектов.

Форма (Form). Форма используется для получения доступа к информации, имеющейся в таблицах. Формы применяются как для ввода новой, так и для просмотра существующей информации.

Отчет (Report). Отчет используется для представления сводных данных. С помощью фильтров и сортировок определяется набор данных, которые должны присутствовать в отчете.

Датапорт (Dataport). Датапорт используется для импорта и экспорта данных в виде текстовых файлов.

XMLport (XMLport). XMLport используется для импорта и экспорта данных формате XML.

Программный модуль (Codeunit). Программный модуль содержит определенные пользователем функции, написанные на языке C/AL. Эти функции можно вызывать из других объектов вашего

DRAFT

приложения. Это сокращает размер приложения, поскольку можно повторно использовать один и тот же код.

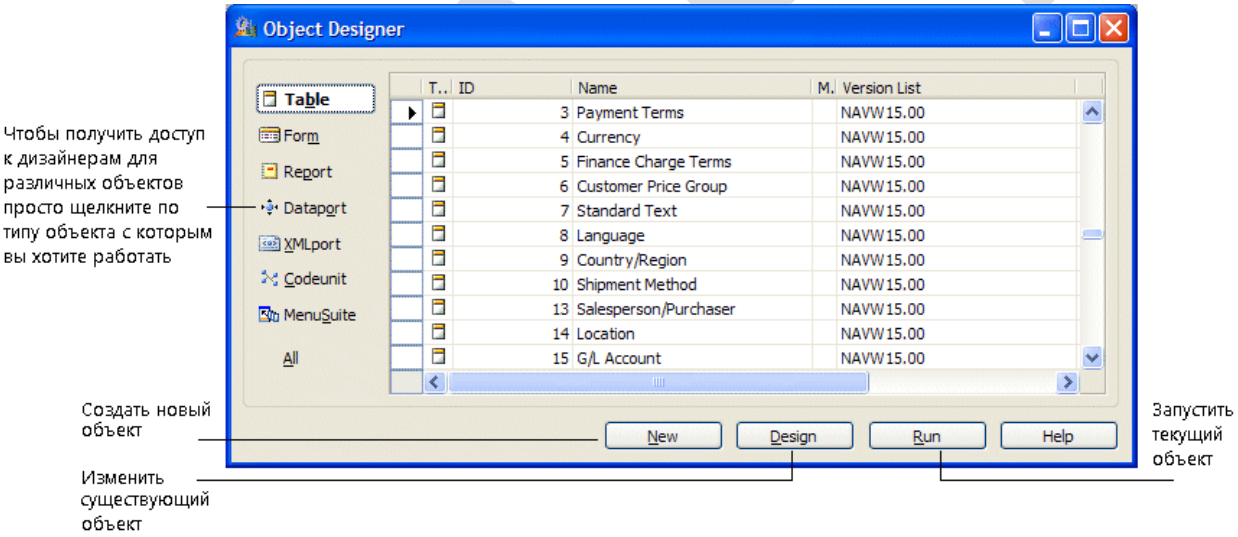
MenuSuite (MenuSuite). MenuSuite содержит набор меню, отображаемых в Панели Навигации.

> **Примечание**
>
> Каждый объект приложения однозначно определяется с помощью идентификатора (ID) Однако существуют ограничения на использование идентификаторов при создании объектов. Более подробную информацию можно получить у сертифицированного партнера Microsoft по направлению Business Solutions.

Object Designer

Основной инструмент, используемый для разработки приложений в среде C/SIDE – это Object Designer, он применяется для:

- Создания новых таблиц, форм, отчетов, датапортов, XMLport, программных модулей. (Для создания объектов MenuSuite используется Дизайнер Панели Навигации)
- Просмотра существующих объектов приложения. Объект MenuSuite можно просматривать либо с помощью Дизайнера Панели Навигации (или непосредственно в Панели Навигации)
- Изменения существующих объектов приложения (кроме объекта MenuSuite, который редактируется с помощью Дизайнера Панели Навигации).
- Запуска объектов приложения (кроме XMLport и MenuSuite).



В следующей таблице представлены доступные в Object Designer инструменты.

Использовать...	...для работы с...
Table Designer	Таблицами
Form Designer	Формами
Report Designer	Отчетами
Dataport Designer	Датапортами
XMLport Designer	XMLport
C/AL Editor	Программными модулями
Дизайнер Панели Навигации	Панелью Навигации

За исключением объекта MenuSuite, одновременно можно работать с несколькими объектами в одном дизайнере. Это значит, что можно одновременно запустить несколько экземпляров дизайнеров. Например, при работе одновременно над тремя новыми формами каждая форма отображается в своем Form Designer. Сам Object Designer может быть запущен только в единственном экземпляре.

Object Designer и объект MenuSuite

Если в Object Designer щелкнуть кнопку MenuSuite, а затем кнопку Design, содержимое объекта будет отображено в Дизайнере Панели Навигации. С его помощью и нужно изменять содержимое объекта.

Щелкните кнопку MenuSuite, а затем кнопку New. Появится диалоговое окно, в котором требуется указать уровень создаваемого объекта MenuSuite (например, Partner или Company). Если для каждого уровня (к которому есть доступ), создан объект MenuSuite, то кнопка New в ObjectDesigner будет недоступна. Причина такого поведения системы заключается в том, что для каждого уровня разрешено создавать только один объект MenuSuite. После того как в окне Уровень Дизайна сделан выбор, откроется Дизайнер Панели Навигации.

Является ли система Dynamics NAV объектно-ориентированной?

Среда C/SIDE является не объектно-ориентированной (object-oriented), а основанной на объектах (object-based). Это важное различие. В объектно-ориентированном языке или инструментальной среде разработчик может создавать новые типы объектов на основе уже имеющихся в системе. В Dynamics NAV вы можете создавать и использовать следующие типы объектов: таблицы, формы, отчеты, датапорты, XMLport, программные модули или MenuSuite. Имея ограничение по количеству объектов, вы можете настроить среду C/SIDE так, чтобы она функционировала быстрее и эффективнее. Кроме того, вы получаете широкие возможности в области проектирования. Вы точно знаете, с чем должны работать. Наибольшее преимущество этого ограничения – стабильность. В среде C/SIDE крайне трудно породить критическую ошибку.

4.2 Что Такое Приложение C/SIDE?

Среда C/SIDE (Client/Server Integrated Development Environment – интегрированная среда разработки в клиент-серверной архитектуре), спроектированая специально с целью создания приложений для бухгалтерского учета и управления бизнесом. Любое приложение C/SIDE состоит из тех же объектов, что и база данных C/SIDE. Разница между терминами база данных и приложение заключается в том, что, когда мы говорим о базе данных, то имеем в виду просто набор прикладных объектов, тогда как под приложением мы понимаем набор взаимосвязанных прикладных объектов, составляющих единое целое.

Общие Принципы Среды C/SIDE

Кроме объекта MenuSuite, остальные типы C/SIDE объектов подчинены общим принципам. Некоторые из этих принципов распространяются на один тип прикладного объекта, другие применимы к нескольким типам. Осознав эти базовые принципы, вы будете иметь прочный фундамент для создания собственных приложений.

В сводной таблице, представленной ниже, показано, как прикладные объекты соотносятся с этими общими принципами, и дается объяснение того, для чего используется каждый тип прикладного объекта.

Тип прикладного объекта	Цель использования	На каких концепциях основан тип

DRAFT

Тип прикладного объекта	Цель использования	На каких концепциях основан тип
Таблица	Таблица используется для хранения фактических данных. Обычно бизнес-приложение содержит таблицу Customer, в которой хранится такая информация о каждом клиенте, как имя, адрес, телефонный номер и контактное лицо.	Свойства, поля, ключи, C/AL код
Форма	Форма используется для получения доступа к информации, имеющейся в таблицах. Формы применяются как для ввода новой, так и для просмотра существующей информации.	Свойства, C/AL код, элементы управления
Отчет	Отчет используется для представления сводных данных. В частности, отчет необходим для вывода на печать списка клиентов.	Свойства, C/AL код, элементы управления, элементы данных, секции, форма запроса
Датапорт	Порт данных используется для импорта данных из других программ и их экспорта в эти программы (например, файлов в формате с запятыми в качестве разделителя для экспорта в электронную таблицу).	Свойства, C/AL код, элементы данных, форма запроса
XMLport	XMLport используется для импорта и экспорта данных в формате XML.	Свойства, C/AL код
Программный модуль	Программный модуль содержит определенные пользователем функции, написанные на языке C/AL. Эти функции можно вызывать из других объектов вашего приложения. Это сокращает размер приложения, поскольку можно повторно использовать один и тот же код.	C/AL код
MenuSuite	MenuSuite содержит меню, отражающиеся в Панели Навигации.	

Термины, содержащиеся в третьем столбце, описаны ниже:

Свойства. Свойства (properties) управляют внешним видом и поведением прикладных объектов, а также всех дочерних объектов. Свойства используются для управления отображением данных, определения значений по умолчанию, цвета и отношений.

C/AL. C/AL – это язык, используемый для написания функций в среде C/SIDE. В приведенной выше таблице термин «C/AL код» относится к функциям, написанным на этом языке.

Триггеры. Триггеры это механизмы, встроенные в объекты приложения. Когда над прикладными объектами совершаются определенные действия, система автоматически активизирует триггер (trigger). Если нужно модифицировать заданное по умолчанию поведение прикладного объекта или расширить его функциональные возможности, в код триггера можно внести собственный код на языке C/AL.

Ключи. Ключ (key) определяет тот порядок, в котором данные хранятся в таблицах. Вы можете ускорить процесс поиска данных в таблицах, определив несколько ключей, которые сортируют данные различными способами.

Поля. Поле (field) – это наименьшая единица информации в базе данных. В поле обычно хранится такая информация, как имя или номер.

DRAFT

Элементы управления. Элементы управления (controls) – это объекты формы или отчета, которые отображают данные, выполняют действия или придают форме привлекательный вид. В качестве типичных примеров можно привести кнопки команд и текстовые метки.

Форма запроса. Форма запроса (request form) – это форма, используемая в отчете. Перед запуском отчета появляется форма отчета, позволяющая пользователю указать необходимые фильтры и опции запроса.

Элементы данных. Элемент данных (data items) – это компоновочный блок, используемый для определения модели данных при создании отчета. Используя иерархию элементов данных, вы определяете, какие данные должны содержаться в отчете. Элемент данных представляет собой таблицу, и когда вы открываете отчет, система последовательно просматривает записи в соответствующей таблице. Элемент данных может содержать одну или несколько секций.

Раздел. Раздел (section) – это подструктура элемента данных. В секцию вы помещаете элементы управления, предназначенные для отображения данных. Обычно используются секции, определяющие содержимое, заголовок и нижний колонтитул в вашем отчете.

4.3 Физическая и Логическая Базы Данных

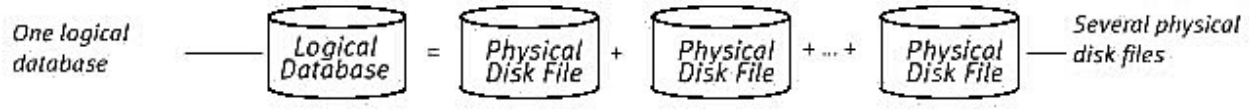
В предыдущем разделе описаны общие принципы, на которых основаны все семь типов прикладных объектов в среде C/SIDE. В этом разделе представлен другой взгляд на приложения C/SIDE. В соответствии с ним нас будет интересовать только то, как структурированы данные в вашем приложении.

Как обычный пользователь базы данных, вы не должны думать о том, где на жестком диске расположена каждая порция данных и каков ее размер. Вы просто хотите быть уверены в том, что, обратившись к имени, получите корректное значение. Поэтому система управления базами данных C/SIDE дает вам концептуальное представление данных, не включающее в себя множество деталей того, как сохраняются данные. Для этого концептуального представления используется абстрактная модель данных. В этой модели данных применяются логические принципы (такие как объекты, их свойства и отношения), которые легче поддаются пониманию.

Таким образом, мы приходим к разграничению логической и физической баз данных. Говоря о логической базе данных, мы интересуемся только структурой данных и отношениями между различными единицами информации. То есть мы не касаемся вопроса того, как эти структуры и отношения реализованы. Говоря о физической базе данных, мы затрагиваем только то, как реализованы структуры логической базы данных и пути поиска в этих структурах.

В рамках данной книги термин «база данных» следует понимать как логическую базу данных, если не указано иное.

То, что пользователю представляется взаимосвязанным набором данных в СУБД C/SIDE, может храниться в нескольких файлах на физическом диске, но это скрыто от пользователя. Представленный ниже рисунок иллюстрирует то, как одна логическая база данных может физически храниться на трех жестких дисках, но, тем не менее, оставаться единой (логической) базой данных.



Логическая Структура Базы Данных

Доступ к данным становится возможным благодаря четко определенной логической организации, состоящей из следующих элементов:

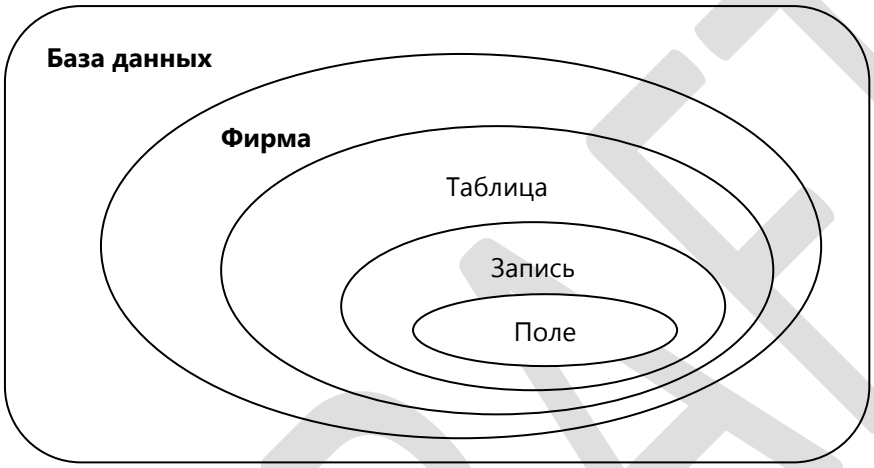
Поля. Поле (field) – это наименьшая логическая структура, используемая в СУБД C/SIDE. Поле используется для хранения одной единицы информации, такой как имя (например, «Евгений») или количество (например, «2 352,00»). Каждое конкретное поле может содержать данные только

одного определенного типа. Поля объединяются в структуру под названием «запись» (record). Само по себе поле не обладает особенной значимостью, так как может содержать лишь ограниченный объем информации. Объединяя эти небольшие единицы информации в записи, мы получаем намного более гибкое «хранилище информации», которое к тому же лучше организовано, поскольку связывает воедино несколько полей.

Записи. Запись (record) – это логическая структура, созданная из произвольного числа полей. Она используется для хранения одной записи в базе данных. Поля записи используются для хранения информации о важных свойствах записи. Записи объединяются в таблицы.

Таблицы. Таблицу (table) можно представить в виде матрицы размером N x M. Каждая строка N описывает запись, а каждый столбец M – поле записи. Таблицы объединяются в фирмы.

Фирмы. Фирма (company) – это наибольшая логическая структура, используемая в СУБД C/SIDE. Фирму можно рассматривать в качестве раздела БД. Ее основная задача – выделять и группировать большие порции данных в базе данных. Фирма может содержать приватные таблицы, а также таблицы, совместно используемые другими фирмами.



DRAFT

ГЛАВА 5 РАЗРАБОТКА ПРИЛОЖЕНИЙ В СРЕДЕ C/SIDE

Тщательное и детальное проектирование базы данных вашего приложения, поможет вам получить стабильный проект. Должным образом спроектированное приложение проще разрабатывать и поддерживать.

В данной главе будут приведены рекомендации по созданию качественных приложений в среде C/SIDE с помощью популярных методик анализа, проектирования и внедрения.

- Введение в разработку приложений в среде C/SIDE

5.1 Введение в Разработку Приложений в Среде C/SIDE

Тщательное и детальное проектирование базы данных вашего приложения, поможет вам быть уверенным в том, что ваша база данных спроектирована наилучшим возможным образом. Должным образом спроектированное приложение проще разрабатывать и поддерживать. В данном разделе будут приведены рекомендации по созданию качественных приложений в среде C/SIDE с помощью популярных методик анализа, проектирования и внедрения.

Разработка приложений в среде C/SIDE обычно включает следующие три шага:

Понимание Задачи. Убедитесь, что вы понимаете решаемую бизнес-задачу. Будьте уверены, что знаете, кто будет пользоваться приложением, и чего они будут пытаться достичь.

Проектирование Таблиц. Начните с разработки модели данных, чтобы определить, как данные хранятся и как наиболее выразительно их использовать. Модель данных определяет:

- Какие таблицы должна содержать база данных.
- Какие типы данных будут храниться в полях каждой таблицы.
- Как данные в таблицах связаны друг с другом
- Ограничения, которые должны гарантировать целостность данных.

Разработка приложений. После проектирования таблиц базы данных, можно приступить к проектированию непосредственно приложения. Проектирования приложения включает:

- Разработка форм (для ввода и получения данных) и отчетов (для просмотра и представления данных).
- Создание C/AL кода, для связи объектов приложения.

Указанные шаги являются взаимозависимыми. При переходе от одного шага к другому возможно вам придется повторно обдумать решения, принятые на предыдущем шаге.

Понимание Задачи

Чтобы решить, какая информация должна храниться в базе данных, следует понять цель, которую должна решить база данных и как база данных будет использоваться. Лучшим способом сделать это является интервью с людьми, которые, будут использовать базу данных. Вовлечение конечных пользователей в данный процесс на раннем этапе, позволит избежать проблемы непонимания цели базы данных. Интервьюирование конечных пользователей поможет понять задачи, которые, по их мнению, должна решать система. Основываясь на данной информации, вы сможете определить необходимые для решения этих задач типы данных, а затем таблицы, формы и отчеты. Чаще всего это является наиболее трудным этапом в процессе проектирования, но при этом самым важным. Возможность использования приложения по назначению зависит от того насколько корректно были спроектированы таблицы, формы и отчеты.

Обсуждения с конечными пользователями даст вам больше информации о выполняемых ими задачах. Таким образом, вы узнаете, какую информацию должны предоставлять формы и отчеты. Однако это может не сильно помочь в проектировании таблиц.

DRAFT

Проектирование Таблиц

Следующим важным этапом является разделение информации, которую требуется хранить в базе данных, на основные категории, такие как клиенты, товары, сотрудники и т.п.

Перед созданием таблиц следуют определиться с моделью данных. Модель данных должна описывать:

- Таблицы в базе данных.
- Поля в таблицах.
- Связи между полями таблиц.
- Ограничения для полей и связей.

Для описания модели данных подходит модель «Сущность-Связь» (ER-модель). ER-модель позволяет преобразовать реально существующие процессы в реляционную базу данных, которой является и среда C/SIDE.

ER-модель разделяет все элементы реальных процессов на две категории: Сущности и Связи. Сущность – независимо существующий реальный или представляемый объект. Связь – может быть объектом, которые существует физически, например машина или человек, а также может быть абстрактным объектом, например компания или работы. Связи показывают, как сущности соотносятся между собой.

Перед использованием ER-модели необходимо выполнить следующие шаги:

1. Определить типы сущностей, связанных с решаемой задачей. Создать таблицы, представляющие каждый тип сущности.
2. Определить свойства каждого типа сущностей и создать поля в таблицах для представления каждого из этих свойств.
3. Определить связи между сущностями и добавить эти связи к таблицам.

Информация, приведенная ниже, не является полным описание всех граней ER-модели. Однако описание модели продемонстрирует преимущества применения формализованного метода проектирования.

Как Принципы ER-Модели Соотносятся с Принципами C/SIDE?

Реальные задачи обычно содержат группы однотипных сущностей. Например, рассмотрим компанию с сотнями клиентов. Все клиенты являются сущностями. У этих сущностей одинаковые наборы свойств, но значения свойств, для каждой сущности свои. Такие схожие сущности обозначаются как тип сущности, который представляет собой не что иное, как нескольких сущностей с одинаковым набором свойств. При применении абстрактной ER-модели в среде C/SIDE, выполняется преобразование абстрактных элементов модели в конкретное представление. Каждый тип сущности связан с таблицей в среде C/SIDE, а каждое свойство – с полем в таблице.

В следующей таблице представлено, как основные элементы ER-модели связаны с принципами C/SIDE.

Элемент ER-модели	Соответствующий принцип C/SIDE
Тип сущности	Таблица
Сущность	Запись
Свойство	Поле

DRAFT

Определение Типов Полей

В ER-модели, после того как были определены все типы сущностей и их свойства, следует определить какие значения могут принимать свойства. В среде C/SIDE это соотносится с определением типов данных для полей в ваших таблицах.

Пример

На основании анализа ER-модели было определено, что вы имеете дело с сущностью описывающей клиентов компании. На основании этого следует создать таблицу Клиент:

Название компании	**Контактное лицо**	**Телефон**	**...**	**Метод оплаты**

Как видно, для таблицы следует создать следующие поля: Название компании, Контактное лицо, Телефон и Метод оплаты. Таким образом, при создании таблицы Клиента, следует выбрать следующие типы данных:

Поле	Описание	Тип данных
Название компании	В поле хранится название клиента (например, «Microsoft Business Solutions ApS»)	Text
Контактное лицо	Контактное лицо в компании (Например, Андрей Панько)	Text
Телефон	Номер телефона клиента (например, +74951234567)	Text
Метод оплаты	Метод оплаты, используемый клиентом (например, «наличными»)	Text

Дополнительную информацию о типах данных в среде C/SIDE см. Выбор Типа Данных.

Значение Ключей в Среде C/SIDE

ER-модель вносит очень важное ограничение для сущностей, принадлежащих определенному типу сущностей (записи в таблицах). Это Ключи или Уникальность для свойств (полей). Тип сущности обычно содержит как минимум одно свойство значения, которого уникально для каждой сущности. Это свойство используется для однозначной идентификации записи. В следующей таблице показано как принципы ER-модели соотносятся с принципами C/SIDE.

Принципы ER-модели	**Принципы среды C/SIDE**
Ограничения для сущностей принадлежащих типу сущности	Ограничения для записей в таблицах
Требование уникальности для свойств сущности	Поля, входящие в первичный ключ в таблице.

Чтобы среда C/SIDE могла эффективно работать с данными в таблицах, они должны быть отсортированы по определенным критериям (по ключу). Например, таблица Сотрудники может

DRAFT

быть отсортирована по номеру карты социального страхования (или табельному номеру), так как этот номер позволяет однозначно идентифицировать каждого сотрудника.

Для поля, которое входит в первичный ключ таблицы действует требование уникальности для всех записей в таблице. Это требование предупреждает появление двух записей содержащих одинаковое значение в поле, входящем в первичный ключ. Это требование не является ограничением для какой-либо конкретной записи, а для всех записей в таблице.

Иногда, ключ может состоять из нескольких полей. В этом случае, комбинация всех значений полей, входящих в первичный ключ должны быть уникальной для каждой записи. Для таблицы можно задать несколько ключей. Дополнительную информацию о создании ключей см. в разделе Как Создавать Первичный Ключ.

Определение Связей

На данном этапе процесса создания приложения уже спроектированы таблицы для хранения информации различных типов. В вашем приложении требуется получать информацию наиболее рациональным способом. Часто ваша база данных будет возвращать на запрос информацию сразу из нескольких таблиц. Среда C/SIDE предоставляет возможность использовать связи между таблицами, содержащими связанную информацию.

В терминологии теории баз данных различают три вида связей:

Связь «Один-ко-Многим» Запись в Таблице 1 может иметь несколько связанных записей в Таблице 2. При этом запись в таблице 2 может быть связана только с одной записью в Таблице 1. Это наиболее часто используемый вид связей в реляционных базах данных.

Связь «Многие-ко-Многим» Запись в Таблице 1 может иметь несколько связанных записей в Таблице 2, запись в Таблице 2 также может иметь несколько связанных записей в Таблице 1. Такая связь сигнализирует о наличие проблемы в проектировании. Обычно связь «Многие-ко-Многим» преобразуют к двум связям вида «Один-ко-Многим».

Связь «Один-к-Одному» Запись в Таблице 1 может иметь только одну связанную запись в таблице 2, запись в Таблице 2 также может иметь только одну связанную запись в таблице 1. Такой вид связей является неэффективным и зачастую его можно избежать, объединив две таблицы.

Гарантия Качественного Проектирования

При определении набора таблиц и создания связей, часто приходится делать выбор между несколькими возможными решениями. Чтобы убедится в том, что выбрано лучшее решение, требуются способы оценки качества проектирования. Следует использовать нормализацию в качестве методики проверки качества проектирования. В процессе нормализации проект базы данных будет подвергнут ряду тестов, чтобы удостоверится, что он соответствует нормальной форме. В теории реляционных баз данных обычно выделяется шесть нормальных форм. Существует достаточно литературы посвященной проектированию реляционных баз данных с описание того, как следует приводить решение к указанным нормальным формам. Для дополнительной информации см. список литературы в конце данной главы.

Разработка Приложений

После того как завершено проектирование таблиц, можно приступить к работе над самим приложением. Этап анализа уже предоставил список вопросов, на которое должно отвечать приложение. Проектирование таблиц дало информацию о том, где и как хранятся данные. Основываясь на данной информации можно приступать к созданию приложения.

Этап разработки приложений включает:

Создание Форм. Формы используются для предоставления или сбора информации. Для создания форм доступен определенный набор элементов, таких как текст, данные, рисунки, линии и цвета.

DRAFT

Создание Отчетов. Отчеты используются для предоставления данных в виде печатных документов. Когда требуется предоставить сводную информацию, отчеты предоставляют больше гибкости, чем формы.

Создания Программных Модулей. Программные модули содержат функции, написанные на языке C/AL. Функции, помещенные в программные модули, можно вызывать из других объектов приложения. Это сокращает размер приложения и упрощает поддержку.

Тестирование и Отладка Приложения. Перед тем как завершить работу над приложением, следует проанализировать его на предмет наличия ошибок проектирования. Обычно это итеративный процесс, после завершения, которого будет получено полезное приложение. Если вы выделили время для тщательного выполнения всех этапов создания приложения, вы получите полностью документированное приложение. Это окажет вам неоценимую помощь, когда впоследствии потребуется внести изменения или дополнительную функциональность в приложение.

Рекомендуемая Литература по Проектированию Баз Данных

Наиболее известными книгами о проектировании баз данных являются:

- C. J. Date. An Introduction to Database Systems. Addison-Wesley Publishing Co.
- Elmasri, R. A. and Navate, S. B. Fundamentals of Database Systems. Benjamin/Cummings.
- Dutka, A. F. and Hanson, H. H. Fundamentals of Data Normalization. Addison-Wesley Publishing Co.
- Michael J Hernandez, Database Design for Mere Mortals. Addison-Wesley Publishing Co.

DRAFT

ЧАСТЬ 3 ТАБЛИЦЫ

ГЛАВА 6 ТАБЛИЦЫ – БАЗОВЫЕ ПОНЯТИЯ

Таблицы – основные объекты любой базы данных. Это истинно независимо от того, какой тип данных необходимо сохранить. Создавая новую базу данных, вы начинаете с построения таблиц. Затем вы создаете формы и отчеты, чтобы получать доступ к данным в таблицах и просматривать их.

В этой главе объясняется, как проектировать необходимые таблицы для хранения данных.

- Что такое таблица?
- Просмотр и модификация свойств
- Создание ключей
- Идентификаторы, типы данных и форматы данных в SQL Server для Dynamics NAV
- Сохранение таблиц и просмотр отсортированных данных
- Специальные поля таблицы
- Разделение таблиц между фирмами

6.1 Что такое таблица?

Записи хранятся в таблицах базы данных C/SIDE. Таблица C/SIDE может быть представлена в виде двумерной матрицы, состоящей из столбцов и строк. На рисунке ниже показана таблица, где каждая строка является записью, а каждый столбец – полем.

DRAFT

Строки: Записи

Столбцы: Поля

G/L Entry - Table

Entry No.	G/L Acco...	Posting D...	D..	Documen...	Description	Bal. Acco...	Amount
1	1110	C31-12-99		START	Opening Entry		1.324.99
2	1140	C31-12-99		START	Opening Entry		-340.59
3	1210	C31-12-99		START	Opening Entry		582.87
4	1240	C31-12-99		START	Opening Entry		-362.26
5	1110	C31-12-99		START	Opening Entry		154.48
6	1140	C31-12-99		START	Opening Entry		-62.92
7	1310	C31-12-99		START	Opening Entry		49.47
8	1340	C31-12-99		START	Opening Entry		-24.80
9	2180	C31-12-99		START	Opening Entry		269.94
10	2180	C31-12-99		START	Opening Entry		135.24

Help

Таблица состоит из двух частей – данных таблицы и ее описания таблицы. Данные таблицы – та часть, которую пользователи часто считают содержимым базы данных, потому что оно состоит из записей со своими полями данных. Однако расположение и свойства этих полей определяются в описании таблицы. Описание таблицы не отображается на экране пользователя. Следующий рисунок иллюстрирует, как данные и описание таблицы формируют таблицу:

Проектируя таблицу, вы присваиваете ей ряд характеристик, таких как имя, идентификационный номер и содержащиеся в ней поля. Кроме того, вы присваиваете каждому полю ряд характеристик (таких как имя, идентификационный номер, тип данных и начальное значение). Проектируя новую таблицу, вы также определяете, какие ключи должна поддерживать система. Сохраняя свой проект таблицы, вы тем самым сохраняете все эти характеристики в описании таблицы.

Информация, хранящаяся в описании таблицы, используется Системой Управления Базой Данных (СУБД) и иногда пользователями базы данных, которым требуется информация о структуре таблицы. Описание таблицы увеличивает гибкость СУБД, т.к. позволяет ей получать доступ к таблицам с различной структурой. СУБД может извлечь описание структуры таблицы из описания таблицы и таким образом корректно подключится к любой таблице.

На следующем рисунке показано, что описание таблицы содержит свойства, триггеры, поля и ключи, а также то, как они связаны между собой:

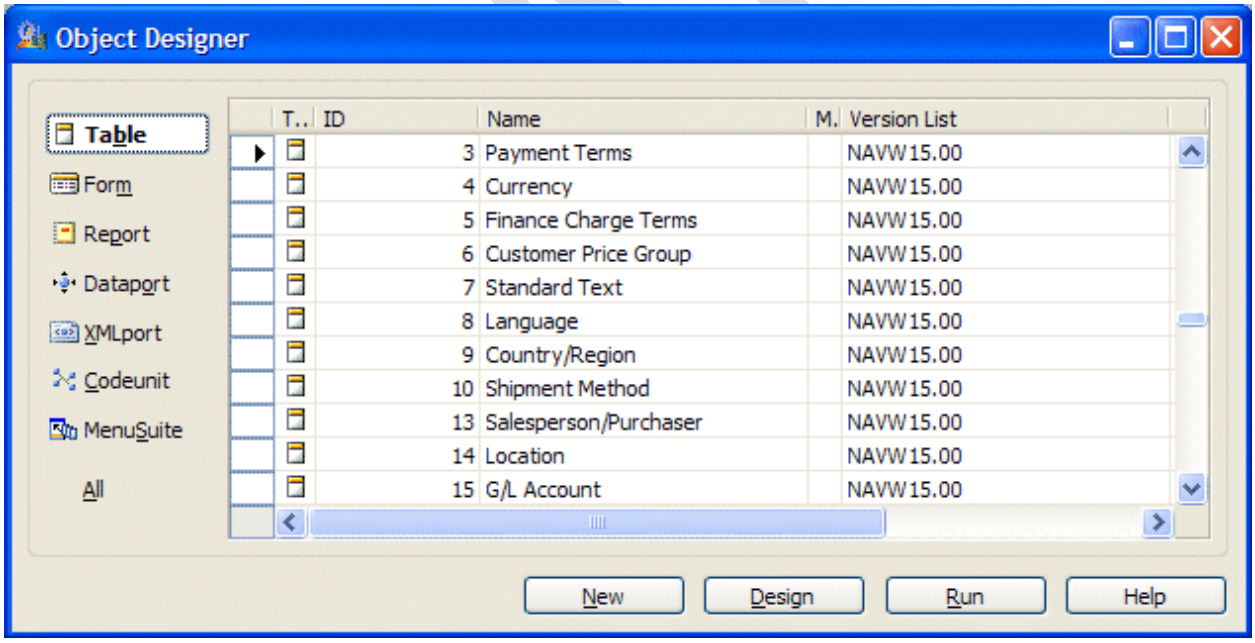
Одни свойства, содержащиеся в описании таблицы, относятся к таблице, а другие – к полям таблицы. Кроме того, есть еще свойства, относящиеся к ключам таблицы. Также можно видеть, что триггеры определены как для таблицы, так и для полей таблицы.

Создание Таблиц

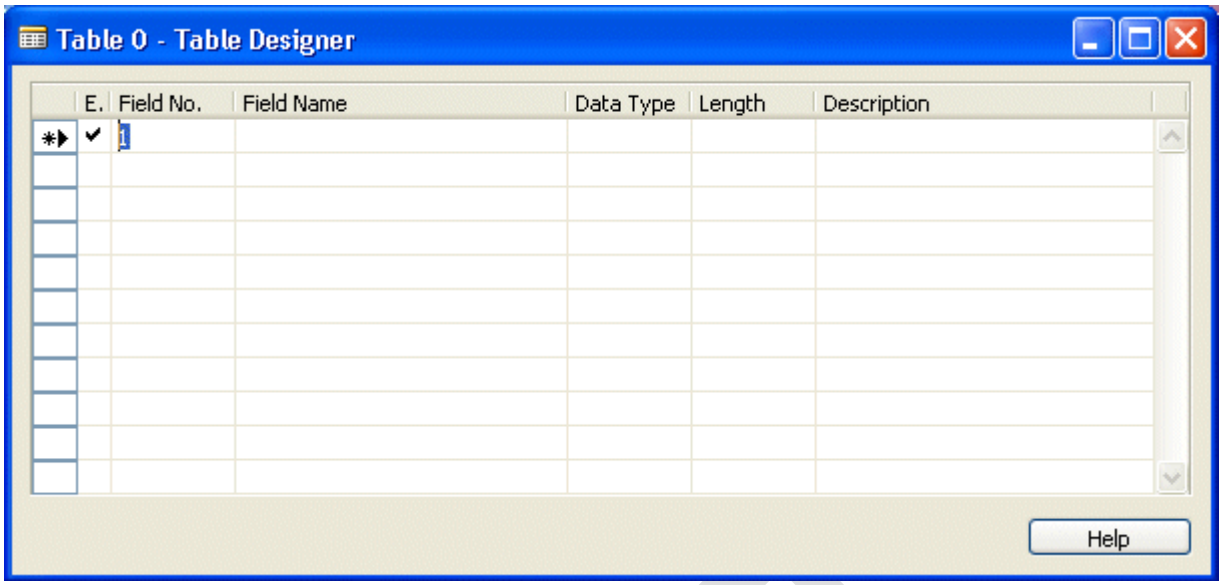
Новая таблица не содержит никаких данных. Создавая таблицу, вы должны решить, какие типы информации вы хотите в ней хранить. Информация хранится в полях, каждое поле может содержать данные одного из поддерживаемых Dynamics NAV типов данных.

Для создания таблицы выполните следующие действия:

1. Щелкните кнопку **Сервис, Object Designer (SHIFT+F12)**. Появится окно **Object Designer**:



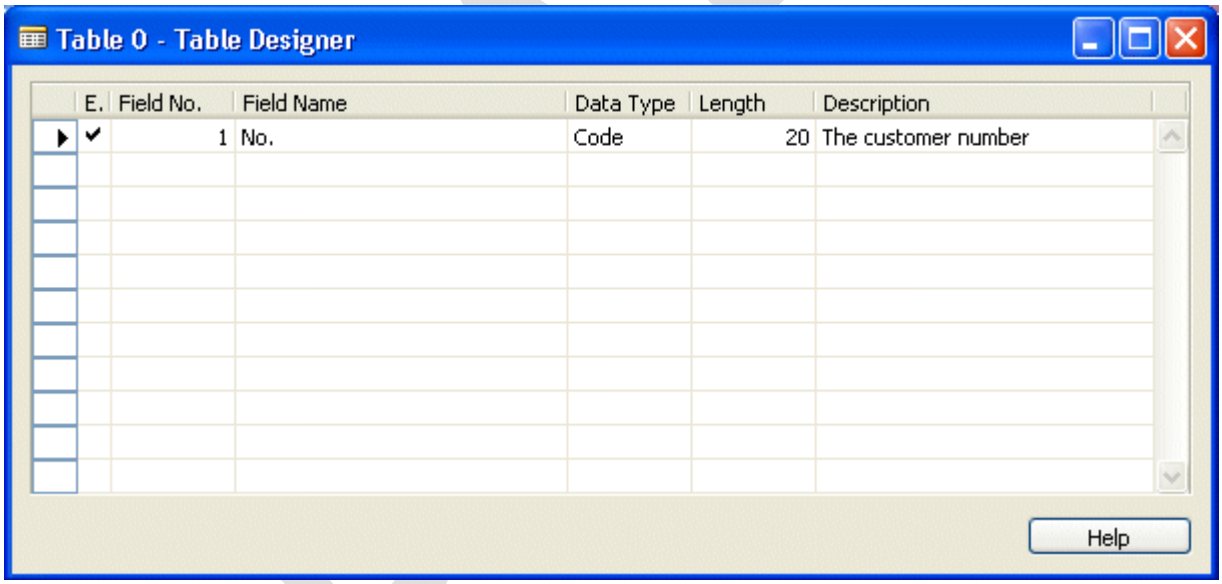
2. Щелкните кнопку **Table**, затем **New**. Появится окно **Table Designer**:



В окне Table Designer введите номер поля, имя, тип данных и, не обязательно, длину и описание для каждого поля, которое вы добавляете к таблице.

Добавление Полей к Таблице

Спроектировать поле – значит присвоить ему ряд характеристик. Эти характеристики зависят от того, в каких целях вы собираетесь использовать поле.



После добавления полей к таблице в окне Table Designer вы должны сохранить таблицу, прежде чем добавлять туда какие-либо записи. После сохранения таблица появится в списке таблиц в окне Object Designer.

Все создаваемые вами таблицы и поля имеют две формы идентификации:

- Уникальный идентификационный номер (целое число). Когда вы обращаетесь к базе данных, используя либо среду C/SIDE, либо C/FRONT, этот номер идентифицирует все таблицы и поля.
- Имя (буквенно-цифровая строка), служащее в качестве надписи (например, КЛИЕНТ или ГОРОД).

это имя появляется на экране, когда вы открываете таблицу, оно должно быть значимым и легко воспринимаемым; данное имя – вторичная информация, и его можно изменить в любой момент.

Выбор Типа Данных

Выбрав идентификационный номер и имя поля, вы должны выбрать соответствующий тип данных. Вы можете использовать различные типы полей в системе баз данных в формате C/SIDE. Каждый тип предназначен для хранения определенного типа данных, таких как текст, числа, даты и так далее.

Поля записей могут быть следующих типов:

Тип Данных	Описание	Размер
Option	Обозначает целое число в диапазоне от –2 147 483 647 до 2 147 483 647. Поле типа Option определяется строкой option, являющейся отделенным запятыми списком строк, представляющих каждое допустимое значение поля. Эта строка используется, когда поле с типом Option отформатировано, а его значение преобразовано в строку. Пример. Поле опции Color определено строкой опции «Красный,Зеленый,Синий». В этом случае допустимые значения поля - 0, 1 и 2, где 0 представляет «Красный» и так далее. При форматировании поля «Color», 0 преобразовывается в строку «Красный», 1 - в «Зеленый», а 2 - в «Синий». Размер соответствующего типа данных SQL – INTEGER - 4 байта. (A)(B)	4 байта
Integer	Обозначает целое число в диапазоне от -2 147 483 647 до 2 147 483 647. Размер соответствующего типа данных SQL – INTEGER - 4 байта. (A)(B)	4 байта
Decimal	Десятичное число в диапазоне между -10^{63} и 10^{63}. Величина экспоненты может быть от -63 до +63. Хранящиеся в памяти десятичные числа содержат 18 значащих цифр Десятичное число представлено в виде BCD (Binary Coded Decimal – двоично-десятичного числа). Размер соответствующего типа данных SQL - DECIMAL (38,20) - 17 байт. (A)(B)	12 байт
Text	Любая буквенно-цифровая строка. Поле должно быть определено так, чтобы оно содержало от 1 до 250 символов. Пространство, используемое текстовым полем, равняется максимальной длине текста плюс один байт. Этот дополнительный байт используется для хранения длины строки. Пустая текстовая строка имеет длину, равную нулю. Размер соответствующего типа данных SQL – VARCHAR – равен 1 байту на символ в значении поля. (A)(B)	Максимальная длина строки + 1 байт (см. примечание ниже).

DRAFT

Тип Данных	Описание	Размер
Code	Буквенно-цифровая строка, выровненная по правому краю, если в ней содержатся только числа. Если среди цифр есть буквы или пробелы, содержание выравнивается по левому краю. Все вводимые символы преобразовываются в верхний регистр. Поле должно быть определено так, чтобы оно содержало от 1 до 250 символов. Пространство, используемое полем Code, равняется максимальной длине текста плюс два байта. Первый из этих дополнительных байтов содержит информацию о длине строки, а второй – о выравнивании. В SQL Server для Dynamics NAV поля Code функционируют по-другому. Вы можете использовать свойство SQL Data Type, чтобы указать, могут ли поля кода содержать целые числа или текстовые строки. Обратитесь к онлайновой справке C/SIDE Reference Guide для получения информации о свойстве SQL Data Type, также смотри ГЛАВА 26 НУМЕРАЦИЯ В DYNAMICS NAV, где содержится информация о сортировке числовых значений в полях Code Размер соответствующего типа данных SQL – VARCHAR – равен 1 байту на символ в значении поля. (A)(B)	Максимальная длина строки + 2 байта (см. примечание ниже).
Date	Значение даты в диапазоне от 1 января 0г. до 31 декабря 9999г. Неопределенная дата - 0. Все даты имеют соответствующую закрывающую дату. Система расценивает закрывающую дату для данной даты как период, который следует за данной датой, но стоит перед следующей нормальной датой. Иначе говоря, закрывающая дата ставится сразу после соответствующей нормальной даты, но перед следующей нормальной датой. Размер соответствующего типа данных SQL – DATETIME – равен 8 байтам. (A)(B)	4 байта
Time	Любое время в диапазоне от 00:00:00 до 23:59:59.999. Поле времени содержит 1 плюс число миллисекунд, начиная с 00:00:00 или 0 (ноль) – неопределенное время. Значение времени рассчитывается следующим образом: Время = 1 + (число миллисекунд, начиная с 00:00:00). Размер соответствующего типа данных SQL – DATETIME – 8 байт. (A)(B)	Поле времени хранится как целое число (4 байта).
Boolean	Принимает значения TRUE или FALSE. После форматирования булево поле представлено значениями «Да» или «Нет». Размер соответствующего типа данных SQL – TINYINT - 1 байт. (A)(B)	4 байта
Binary	Содержит двоичные данные. Двоичные данные хранятся в записи. Размер соответствующего типа данных SQL – VARBINARY – равен числу байт в значении поля. (A)(B)	Максимальная длина - 250 байт (см. примечание ниже).

DRAFT

Тип Данных	Описание	Размер
BLOB	Binary Large Object – «большой двоичный объект». Используется для хранения растровых изображений и текстов. Обратите внимание, что BLOB хранится не в записи, а в BLOB-области таблицы. Размер соответствующего типа данных SQL – IMAGE – равен числу байт в значении поля. (A)(B)	8 байт в записи + размер BLOB-данных (макс. 2 Гбайт)
DateFormula	Используется для хранения формулы даты. Пример синтаксиса: 30D (=30 дней) CM+1M (=текущий месяц плюс один месяц) D15 (=пятнадцатое число каждого месяца)	4 байта
TableFilter	Этот тип данных используется для применения фильтров к другим таблицам. В настоящее время, используется для хранения Фильтров Защиты в таблице Права Доступа (Permission)	
BigInteger	Целое размером 64 бит	8 байт
Duration	Представляет в миллисекундах длительность между двумя моментами времени. Значение может быть отрицательным	8 байт
DateTime	Представляет момент времени как комбинацию даты и времени. Datetime хранится в базе данных как универсальное глобальное время (Coordinated Universal Time - UTC). В Dynamics NAV всегда отображается как локальное время. Локальное время определяется на основании часового пояса, указанного в региональных настройках компьютера. Значение в поле типа DateTime всегда должно вводиться как локальное время. При вводе локального времени, система конвертирует его в UTC, использую текущие настройки часового пояса и перехода на зимнее время. DateTime не поддерживает закрывающие даты.	Хранится как два integer по 4 байта
GUID	Глобально уникальный идентификатор (Globally unique identifier)	16 bytes
RecordID	Уникальный идентификатор записи	

(A) Вычисление размера определенной записи SQL Server заключается не просто в сложении размеров значений полей. Обратитесь к документации по SQL Server корпорации Microsoft для получения дополнительной информации.

(B) Это тип данных SQL Server, который использует система Dynamics NAV, создавая тип данных Dynamics NAV. Для более подробной информации см. Представление Типов Данных Dynamics NAV.

Примечание

В C/SIDE Database Server данные хранятся кратно четырем байтам для повышения производительности. Размер текста, кода и двоичных полей (которые могут иметь различную длину) округляются до ближайшего значения, которое кратно четырем. Это означает, что, например, текстовая строка из 10 символов займет 12 байт.

Кроме обычных полей, описанных в данном разделе в среде C/SIDE, есть еще два специальных типа полей:

- Вычисляемые Поля (FlowField)

DRAFT

- Вычисляемые Фильтры (FlowFilter)

Преимущества, предоставляемые указанными полями при получении данных, будет описано в разделе 6.6 Специальные Поля Таблиц.

6.2 Просмотр и Модификация Свойств

В данном разделе описано, как использовать свойства при проектировании таблиц. Как вы узнали ранее, существует три вида свойств:

- Свойства таблиц
- Свойства полей
- Свойства ключей

Просмотр и Модификация Свойств Таблиц

Таблица в среде C/SIDE обладает рядом свойств, которые описывают поведение таблицы в определенном вычислительном окружении. Когда вы создаете таблицу, среда C/SIDE автоматически определяет ряд значений этих свойств по умолчанию. В зависимости от цели таблицы и от того, как она относится к другим объектам-приложениям, вам может потребоваться изменение этих значений по умолчанию.

В среде C/SIDE существуют следующие свойства таблиц:

Наименование Свойства	Данное свойство используется для …
ID	Определяется идентификатор таблицы
Name	Определяется имя (используется как заголовок) для таблицы.
Caption	Отображается как заголовок на текущем выбранном языке. Значение берется из свойства *CaptionML,* если оно задано. Заголовок – это текст, который система использует для отображения обозначения элемента управление (например, в строке заголовка формы или в метке для других элементов управления)
CaptionML	Предоставляет текст, который используется системой для отображения обозначения элемента управления или иного объекта в пользовательском интерфейсе. Поле CaptionML поддерживает многоязычность. Это значит, что оно может содержать список текстовых строк на разных языках. Данные текстовые строки используются в зависимости от выбранного пользователем языка.
Description	В поле содержится необязательное описание таблицы. Это поле используется исключительно для внутренних целей и не видно конечному пользователю. Краткое описание назначения таблицы поможет упростить поддержку приложения.
DataPerCompany	Определяется, будет ли система создавать по отдельной версии данных для каждой фирмы в базе данных.
Permissions	Определяются расширенные права доступа для таблицы.
LookupFormID	Указывается идентификатор формы, которая будет использоваться подбора.
DrillDownFormID	Указывается идентификатор формы, которая будет использоваться для детализации.
DataCaptionFields	Указывается список полей, которые будут отображаться в качестве заголовка, когда запись из данной таблицы будет отображаться, например, на форме.

DRAFT

Наименование Свойства	Данное свойство используется для …
PasteIsValid	Указывает системе, разрешено ли в данную таблицу добавлять записи вставкой.
LinkedObject	Указывается, связана ли данная таблица Dynamics NAV с существующим объектом SQL Server.
LinkedInTransaction	Определяет, будет ли связанный объект поддерживать транзакции и будет ли доступен в пределах транзакции Dynamics NAV или не будет поддерживать транзакции. Это свойство доступно, только если в свойстве *LinkedObject* установлено значение Да. Дополнительную информацию см. в разделе 7.4 Связанные Объекты (Linked Objects).

Дополнительную информацию о данных свойствах см. в онлайновой справке *C/SIDE Reference Guide*.

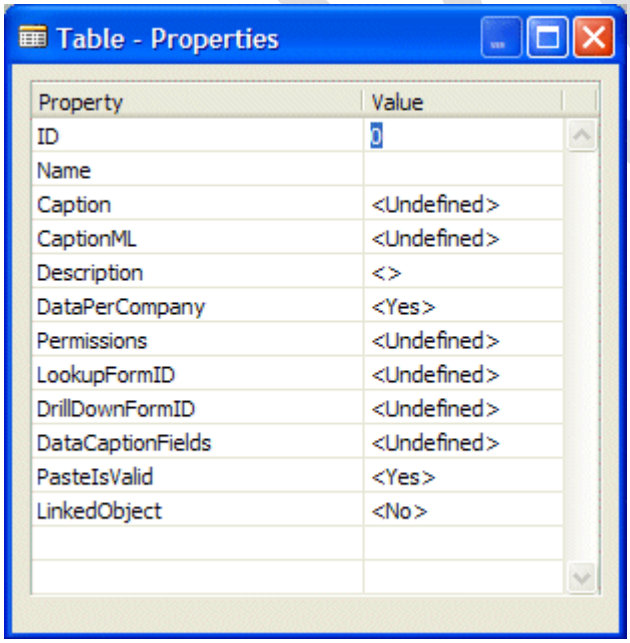
Для просмотра или изменения свойств таблицы выполните следующие действия:

1. Щелкните кнопку Сервис, Object Designer (SHIFT+F12). Откроется окно ***Object Designer,*** щелкните кнопку Table, чтобы увидеть список таблиц.

2. Выберите таблицу и щелкните кнопку Design. Среда C/SIDE отобразит таблицу в окне Table Designer.

3. Установите курсор в пустой строке в Дизайнере таблиц или нажмите кнопку Правка, Выбрать Объект. (Если вы установите курсор в строке, определяющей одно из полей таблицы, то получите свойства данного поля, а не таблицы).

4. Щелкните кнопку Вид, Properties (SHIFT+F4). Среда C/SIDE отобразит окно ***Properties***:



5. Если вы хотите изменить значение свойства, просто введите новое значение в окне Properties. После ввода нового значения нажмите клавишу ENTER или просто уберите курсор с поля, чтобы обновить свойство.

6. Для получения справки по свойству укажите на него в окне Properties и нажмите клавишу F1

Пример

Чаще всего требуется изменить значение свойства LookUpFormID. Значение по умолчанию свойства *LookUpFormID* - *<Undefined>*. Изменив данное свойство можно указать какую форму должна открывать система при нажатии на кнопку Выбор (F6). Посмотрите на значение свойства *LookupFormID* в таблице ***Customer*** (18). Данное свойство указывает среде C/SIDE, какую форму должна использовать система при выборе из таблицы ***Customer***. Значение данного свойства - форма ***Customer List***.

Просмотр и Модификация Свойств Полей

Как и в случае с таблицами, все поля в среде C/SIDE имеют набор свойств, которые определяют их поведение. Когда вы создаете поле, среда C/SIDE автоматически предлагает ряд значений этих свойств по умолчанию. В зависимости от цели поля, вам иногда придется изменять эти значения, назначенные по умолчанию.

В среде C/SIDE существуют следующие свойства полей:

Название Свойства	Данное поле используется для...
Field No.	Присваивается уникальный идентификатор для поля.
Name	Определяется название поля.
Caption	Заголовок текстового поля.
CaptionML	Заголовок текстового поля, которое используется системой для идентификации элемента управления или иного объекта в пользовательском интерфейсе. Свойство CaptionML является многоязыковым, т.е. содержит список текстовых строк на различных языках.
CaptionClass	Позволяет управлять полем таблицы или элементом управления с помощью caption classes.
Description	Включает необязательное описание поля. Данное описание используется исключительно для внутренних целей и не видно конечному пользователю.
Data Type	Указывается тип данных поля таблицы
Enabled	Указывается, включено ли поле
Data Length	Указывается максимальное значение длины данных, хранимых в данном поле.
InitValue	Указывается значение поля, которое устанавливается при инициализации записи
FieldClass	Указывается класс поля (здесь указывается, является ли поле Нормальным полем, Вычисляемым Полем или Вычисляемым Фильтром).
CalcFormula	Задается формула, используемая Вычисляемым Полем (FlowField).
AltSearchField	Указывается поле, используемое для альтернативного поиска.
DecimalPlaces	Указывается количество знаков после запятой, отображаемых пользователю. Также данное свойство используется для проверки значений, введенных пользователем, на соответствие настройкам данного свойства.
Editable	Указывается можно ли редактировать данное поле.

DRAFT

Название Свойства	Данное поле используется для…
NotBlank	Указывается, что пользователь не может оставить данное поле незаполненным
BlankNumbers	Указывается, какие значения, введенные в числовое поле, система должна скрывать
Numeric	Принуждает пользователя вводить в данное поле только цифры
CharAllowed	Набор символов разрешенных для ввода в данное поле.
DateFormula	Проверяет синтаксис выражения по расчету даты, введенному пользователем (применяется для полей типа Text и Code, использовать не рекомендуется т.к. для хранений формул расчета даты используется специальный тип данных DateFormula).
Standard day/time unit	Указывается, какая используется единица измерения при вводе данных в поле с типом данных Duration.
MinValue	Указывается минимальное разрешенное значение в поле
MaxValue	Указывается максимальное разрешенное значение в поле
Title	Первая буква в каждом слове будет приведена в верхний регистр.
ValuesAllowed.	Указываются значения, разрешенные для ввода в данное поле
AutoIncrement	Указывает, должно ли полю автоматически присваиваться новое числовое значение, которое больше числового значения в предыдущем поле.
TableRelation	Указываются связи с другими таблицами. Смотри 7.2 Создание Связей между Таблицами для дополнительной информации.
ValidateTableRelation	Указывается, должна ли система проверять введенное значение с учетом связи между таблицами.
TestTableRelation	Указывается, должно ли данное поле включаться в процедуру тестирования связей между таблицами.
TableIDExpr	Указывается идентификатор таблицы, к которой будет применен фильтр таблиц.
BlankZero	Указывается, что поля, содержащие значения 0 (ноль) или Нет будут отображаться как пустые, если в данном свойстве установлено значение Да.
DataLength	Задается длина данных, которые можно хранить в поле.
OptionString	Определяется строка option (строка, содержащая возможные для выбора варианты, разделенные запятыми). Максимальная длина строки 1000 символов.
ClosingDates	Разрешает ввод закрывающих дат.
AutoFormatType	Определяет форматирование данных.
AutoFormatExpr	Определяет форматирование данных.
SignDisplacement	Смещает отрицательные значения вправо для целей отображения на экране.
SQLDataType	Указывается тип данных, который разрешен для полей типом данных Code. Данное поле используется для полей с типом данных Code при работе с SQL Server для Dynamics NAV.

DRAFT

Название Свойства	Данное поле используется для...
ClearOnLookup	Указывает системе, что текущее значение поля должно быть удалено перед тем как будет добавлено значение введенное пользователем с помощью кнопки Выбор (F6).
SubType	Задается подтип для BLOB-полей, например, Bitmap или Memo.
Compressed	Указывается, должны ли сжиматься BLOB. Данное свойство используется только для BLOB-полей при работе с SQL Server.
OptionCaption	Определяются заголовки строки option, которые будут показываться конечному пользователю при выборе значения из поля Option.
OptionCaptionML	Определяются заголовки строки option, которые будут показываться конечному пользователю при выборе значения из поля Option *OptionCaptionML* используется вместе со свойством *OptionString*. Свойство *OptionString* содержит набор значений доступных для выбора, эти значения хранятся в базе данных или используются в коде, написанном на языке C/AL.

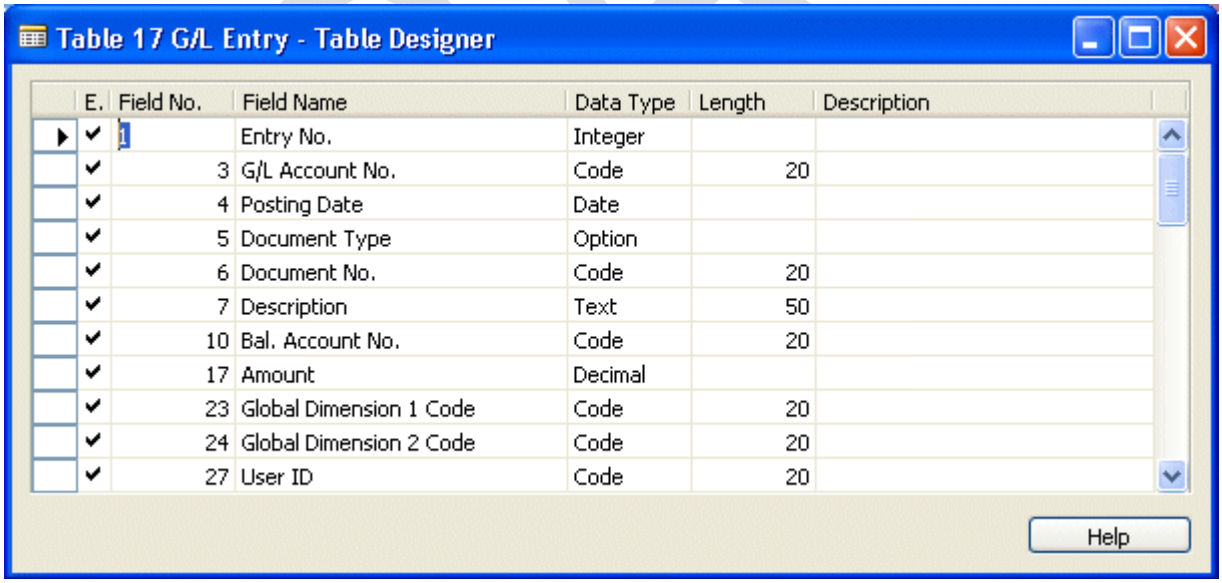
Дополнительную информацию о данных функциях смотрите в онлайновой справке *C/SIDE Reference Guide*.

Для просмотра и изменения свойств полей выполните следующие действия:

1. Щелкните Сервис, Object Designer (SHIFT+F12).

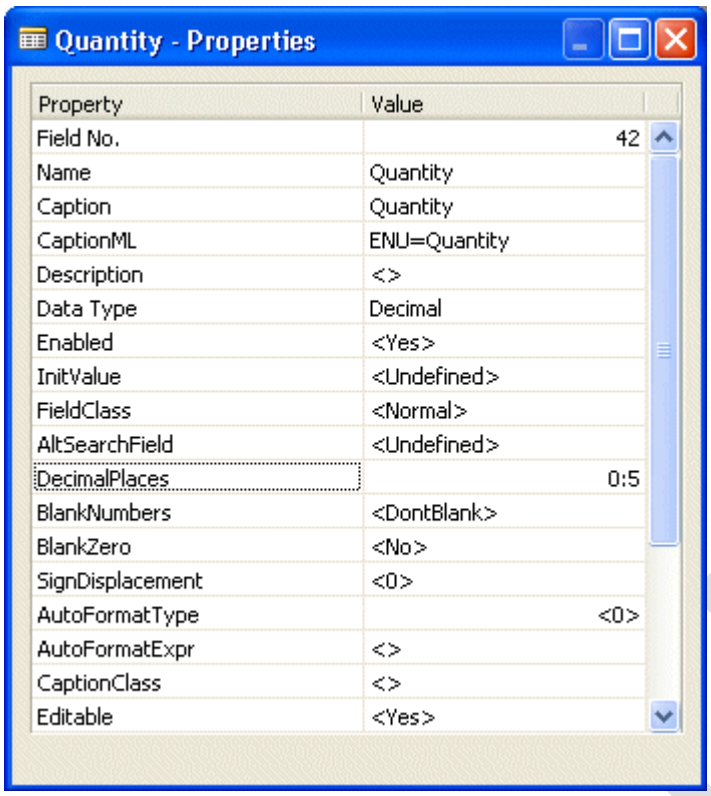
2. В окне ***Object Designer*** щелкните кнопку **Table**, чтобы открыть список таблиц.

3. Выберите таблицу и щелкните кнопку **Design**, для открытия выбранной таблицы в **Table Designer**:



4. Установите курсор в строке **Table Designer**, содержащей поле, для которого вы хотите получить доступ к свойствам.

5. Нажмите кнопку Вид, Properties. C/SIDE отобразит окно **Properties**:



6. Если вы хотите изменить значение свойства, просто введите новое значение в окне **Properties**. После ввода нового значения нажмите клавишу ENTER или просто уберите курсор с поля, чтобы обновить свойство.

7. Для получения Справки по свойству укажите на него в окне Properties и нажмите клавишу F1.

Пример

Часто требуется изменить значение свойства *DecimalPlaces*. При создании нового поля с типом данных Decimal, среда C/SIDE предполагает, что значения в данном поле должны форматироваться как суммы в денежном выражении. Если в данном поле не хранятся значения сумм денег, вы можете изменить количество отображаемых на экране знаков после запятой. Например, в таблице Фин. Книга Операций (G/L Entry), свойство *DecimalPlaces* поля Количество (Quantity, Field No. 42) установлено как 0:5. Это значит, что минимальное разрешенное количество знаков после запятой – 0, а максимальное 5.

6.3 Создание Ключей

Система управления базами данных (СУБД) получает доступ к каждому полю на основании номера поля и значения первичного ключа записи. В первичный ключ может входить до 20 полей в записи. Комбинация значений в полях в основном ключе позволяет СУБД выполнять уникальную идентификацию каждой записи. Первичный ключ определяет логический порядок, в котором хранятся записи, независимо от их физического размещения на диске.

Логически записи хранятся последовательно в порядке возрастания и сортируются в соответствии с первичным ключом. Перед добавлением новой записи в таблицу СУБД проверяет, чтобы информация в полях первичного ключа в записи была уникальна, и только потом вставляют запись в нужную логическую позицию. Поскольку записи сортируются «на лету», структура базы данных всегда будет правильной. Это позволяет быстро обрабатывать и получать данные.

Описание таблицы содержит список ключей. Ключ (key) – это последовательность одного или более идентификаторов полей из таблицы. С таблицей может быть связано до 40 ключей. Первый ключ в списке – первичный (primary key).

DRAFT

Первичный ключ всегда активен. СУБД поддерживает сортировку таблицы в соответствии с порядком первичного ключа и отклоняет записи с одинаковыми значениями в полях первичного ключа. Поэтому значения в первичном ключе должны всегда быть уникальными. Помните, что при использовании составного ключа, уникальной должна быть комбинация значений во всех полях, из которых состоит первичный ключ.

Система баз данных в формате C/SIDE не поддерживает таблицы без ключей.

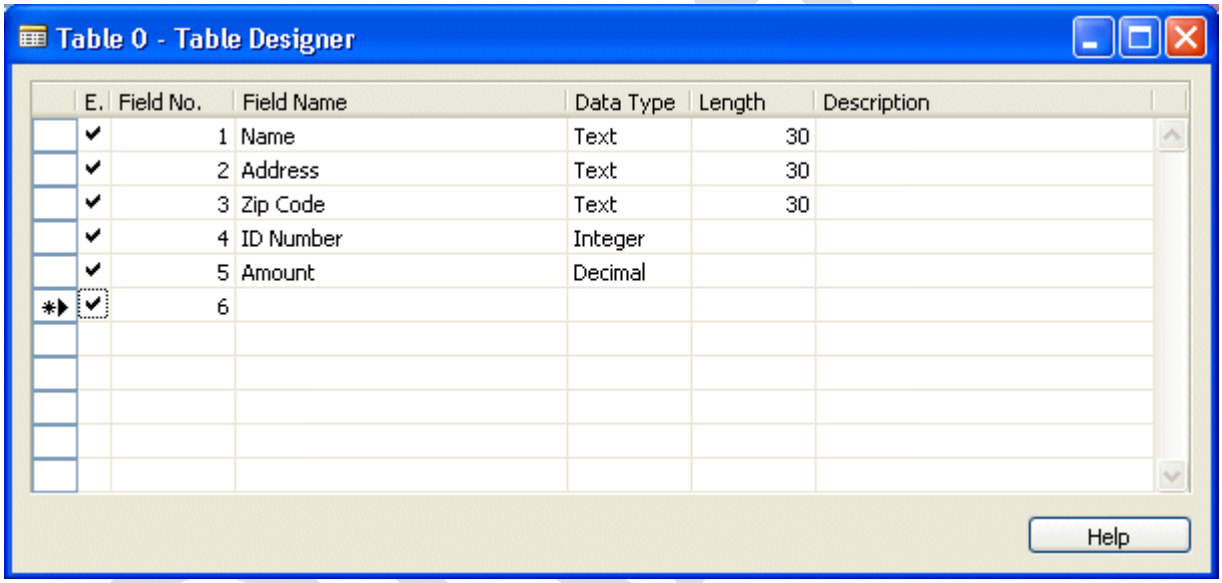
Как Создавать Первичный Ключ

В первичный ключ может входить максимум 20 отдельных полей. Количество полей в первичном ключе ограничивает количество полей в других (вторичных) ключах.

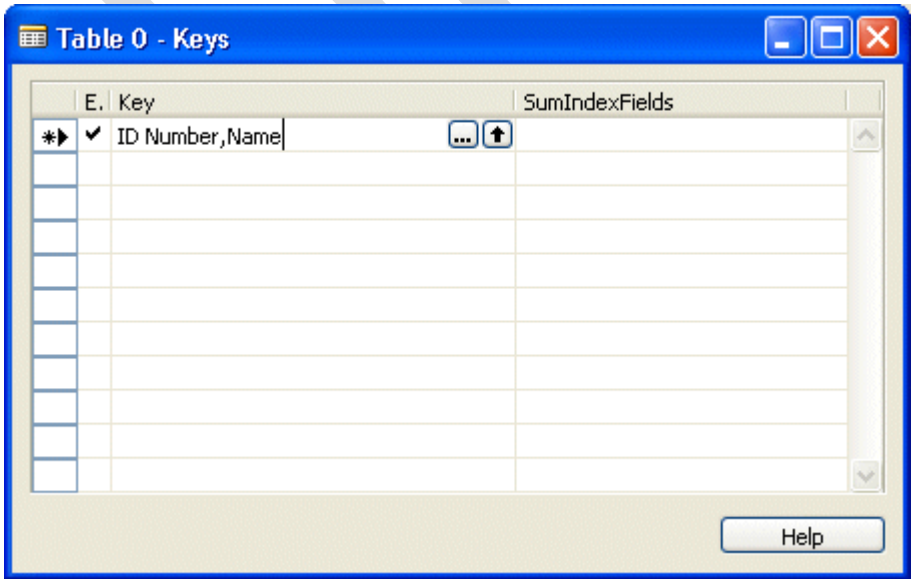
При создании таблицы в table designer, среда C/SIDE автоматически использует поле с наименьшим номером поля в качестве первичного ключа.

Чтобы создать первичный ключ:

1. Предположим, что в Table Designer создана следующая таблица:



2. Щелкните кнопку Вид, Keys чтобы создать первичный ключ. C/SIDE отобразит окно ***Keys***:



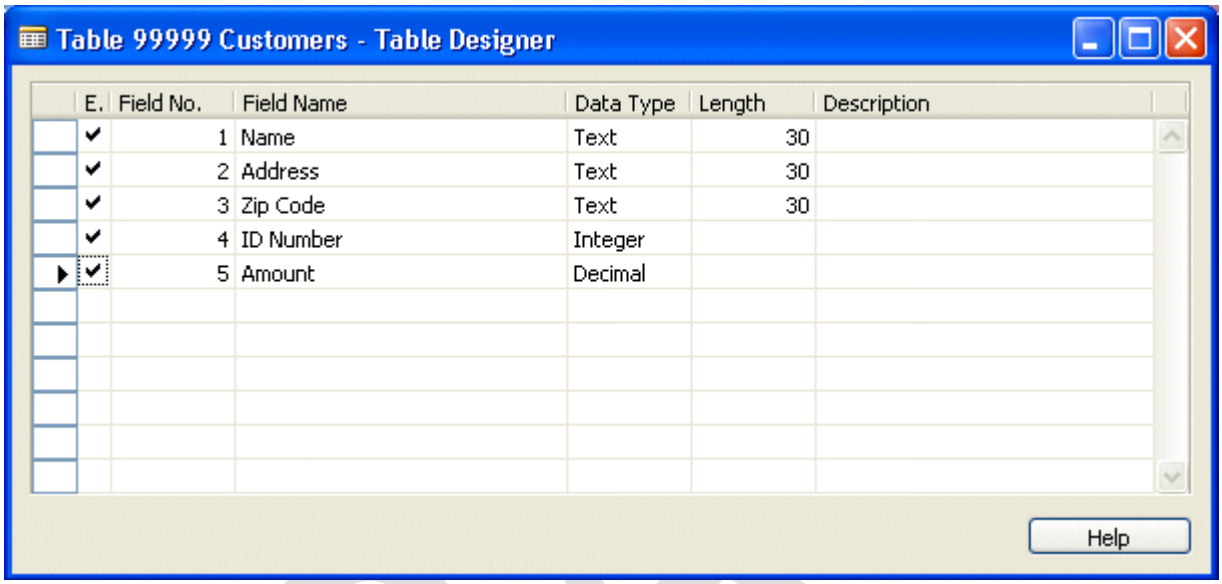
3. В первой строке окна Keys введите первичный ключ, разделив входящие в него поля запятыми (например, ID Number,Name).

Как Создать Вторичный Ключ

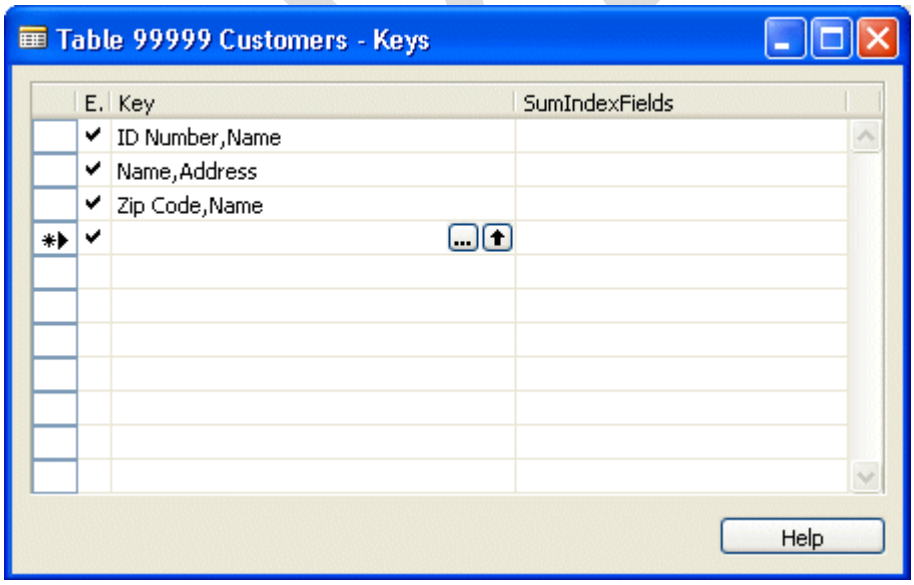
Как было упомянуто выше, с таблицей может быть связано до 40 ключей, и что первый ключ называется первичным. Все остальные ключи – вторичные и не обязательные. Вторичные ключи используются для просмотра записей в порядке, отличном от того, в котором они сортируются согласно полям первичных ключей.

Чтобы создать вторичный ключ:

1. Щелкните Сервис, Object Designer (SHIFT+F12), затем откройте вашу таблицу в Table Designer:



2. Щелкните Вид, Keys. C/SIDE отобразит окно ***Keys*** для таблицы:



3. Первая строка содержит первичный ключ. Введите вторичные ключи в последующие строки, разделив входящие в них поля запятыми (например, Name,Address).

> **Примечание**
>
> Число полей в первичном ключе вместе со всеми полями в каждом вторичном ключе не должно превышать 20. Это означает, что, если ваш первичный ключ включает в себя четыре определенных поля, то вторичный ключ может содержать эти четыре поля и максимум 16

других. Соответственно, если первичный ключ состоит из 20 определенных полей, то вторичные ключи должны состоять только из комбинаций этих полей.

Вторичный ключ использует дополнительную структуру, называемую индексом (index). Эта идея аналогична предметному указателю в книге. В предметном указателе представлен список наиболее важных терминов, отсортированных по алфавиту. Рядом с каждым термином указаны номера страниц, где он упоминается. Предметный указатель может использоваться для быстрого поиска списка страниц (адресов), на указанных страницах легко можно найти требуемый термин. Предметный указатель точно указывает, где в книге используется каждый термин.

Когда вы определяете вторичный ключ и помечаете его как активный, система автоматически обновляет индекс, отражающий порядок сортировки, определенный ключом. Одновременно может быть активно несколько вторичных ключей.

Вторичный ключ можно сделать неактивным. Это означает, что СУБД не обновляет индекс во время обновлений. Кроме того, неактивный ключ не занимает пространство базы данных. Неактивные ключи можно снова активизировать. На выполнение этого процесса может потребоваться некоторое время, величина которого зависит от размера таблицы потому, что СУБД должна просмотреть всю таблицу, чтобы восстановить индекс.

Поля, включающие в себя вторичные ключи, не обязательно содержат уникальные данные. СУБД не отклоняет записи с одинаковыми данными в полях вторичных ключей. Если две или несколько записей содержат идентичную информацию во вторичном ключе, СУБД будет использовать первичный ключ для таблицы, чтобы разрешить этот конфликт.

Порядок Сортировки и Вторичные Ключи

В следующем примере будет показано, как первичный ключ влияет на сортировку при активации вторичного ключа.

Предположим, что таблица ***Customer (Клиенты)*** содержит четыре записи. Записи в таблице ***Customer*** состоят из двух полей: **Customer No. (Клиент Но.)** и **Customer Name (Клиент Название)**.

Ключи таблицы ***Customer***:

Ключ Но.	Тип Ключа	Поля, входящие в ключ
1	Первичный	Customer No.
2	Вторичный	Customer Name

Таблица ***Customer***, отсортированная по первичному ключу:

Customer No.	Customer Name
001	Microsoft Business Solutions
002	IBM
003	Lotus
004	Microsoft Business Solutions

Если для сортировки выбрать вторичный ключ, порядок сортировки будет определяться на основании содержания поля **Customer Name**. Так как содержимое данного поля не является уникальным, для записей будет применена дополнительная сортировка по первичному ключу.

Customer Name	Customer No.
IBM	002

DRAFT

Customer Name	Customer No.
Lotus	003
Microsoft Business Solutions	001
Microsoft Business Solutions	004

В данном случае, две последние записи, у которых одинаковые значения в поле Customer Name, были отсортированы по полю Customer No.

Как Ключи Влияют на Производительность

Обычно поиск выполняется быстрее, если для таблицы, содержащей требуемые данные, создано и поддерживается несколько ключей. Индексы, созданные для каждого ключа, позволяют выполнять быстрый и гибкий поиск. Однако при использовании нескольких ключей кроме преимуществ существуют и отрицательные стороны. Рассмотрим следующие ситуации:

Если	Производительность увеличится	Производительность снизится
Увеличить число активных вторичных ключей.	При получении данных при использовании различных сортировок, т.к. данные в системе уже отсортированы.	При вводе данных, т.к. C/SIDE вынужден поддерживать каждый вторичный ключ.
Решено использовать малое количество ключей.	При вводе данных, т.к. C/SIDE должен поддерживать минимальное количество индексов.	При получении данных. Возможно, придется создать или активизировать вторичный ключ, чтобы получить данные с подходящей сортировкой. В зависимости от размера базы данных эта процедура может потребовать некоторое время, так как система должна создать индекс.

Это сложное решение – определить оптимальное количество используемых ключей. Выбор подходящих ключей и количества активных ключей должен определяться как компромисс между максимальной скоростью получения данных и обновления данных (операции вставки, изменения и удаления данных). Часто, бывает полезно отключить сложные, редко используемые ключи.

В целом, скорость работы среды C/SIDE зависит от ряда факторов:

- Размера вашей базы данных
- Количества активных ключей
- Сложности ключей
- Числа записей в таблицах
- Скорость компьютера и его дисковой системы

Как Хранятся Ключи

Как было показано на рисунке в разделе 6.1 Что такое таблица?, ключи хранятся в Описании Таблицы, в которой содержится список ключей. На следующем рисунке будет показана часть списка ключей для таблицы ***Cust. Ledger Entry (Клиент Книга Операций)***.

DRAFT

1 (Entry No.)				Первичный Ключ
3 (Customer No.)	4 (Posting Date)			Вторичный Ключ
5 (Document Type)	6 (Document No.)	3 (Customer No.)		Вторичный Ключ
3 (Customer No.)	36 (Open)	43 (Positive)	37 (Due Date)	Вторичный Ключ

На рисунке показаны первые четыре ключа данной таблицы: первичный ключ и три вторичных ключа. Первичный ключ состоит из одного поля. Первый вторичный ключ состоит из двух полей. В тоже время второй и третий вторичные ключи состоят из трех и четырех полей соответственно.

Просмотр и Модификация Свойств Ключа

Связанный с таблицей ключ имеет ряд свойств, которые описывают его поведение, аналогично таблицам и полям таблиц. При создании ключа, среда C/SIDE заполняет рад свойств значениями по умолчанию. В зависимости от предназначения ключа, может потребоваться изменить эти значения на более подходящие.

В среде C/SIDE существуют следующие свойства ключей:

Наименование Свойства	Данное свойство используется для …
Enabled	Указывается, будет ли система поддерживать индекс для данного ключа. Если значение данного свойства установлено Нет, ключ нельзя использовать.
Key	Задается ключ.
SumIndexFields	Определяются поля, по которым система должна поддерживать SumIndex.
KeyGroups	Указывается, членом какой группы ключей является данный ключ. Включив ключ в предопределенную группу ключей, можно включать ключ только перед тем как он будет использован.
BackupKey	Показываются ошибки, произошедшие при восстановлении базы данных из резервной копии.
MaintainSQLIndex	Указывается, должны ли для SQL Server создаваться индексы соответствующие ключам Dynamics NAV.
MaintainSIFTIndex	Указывается, должны ли для SQL Server создаваться SIFT-структуры соответствующие SumIndexFields ключа Dynamics NAV.

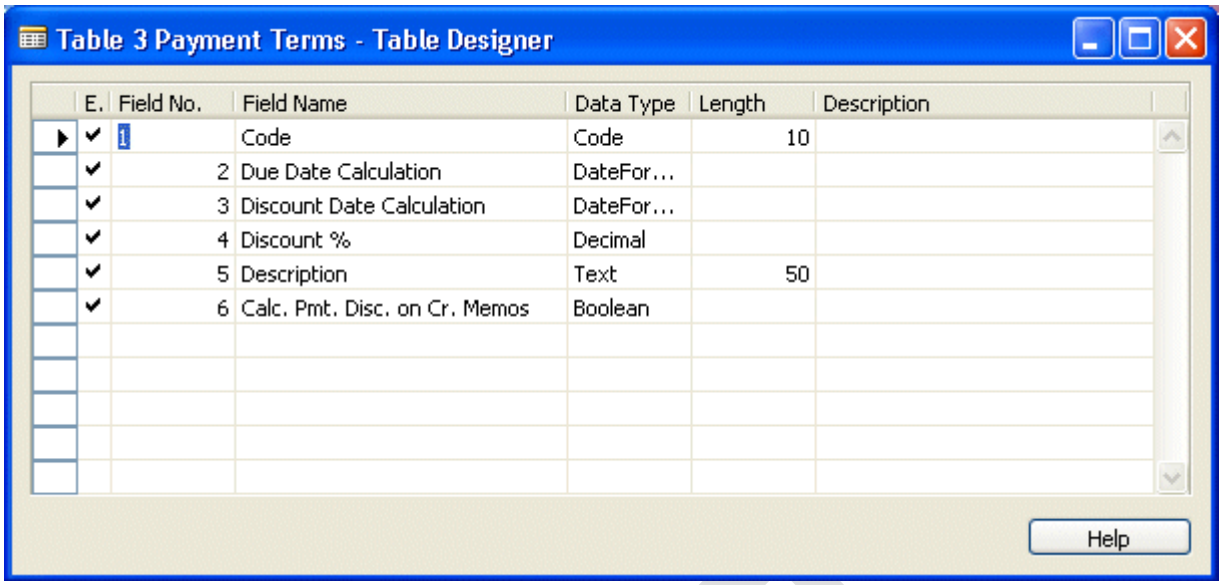
Дополнительную информацию о данных свойствах см. в онлайновой справке *C/SIDE Reference Guide*.

Для просмотра или изменения свойств ключей конкретной таблицы выполните следующие действия:

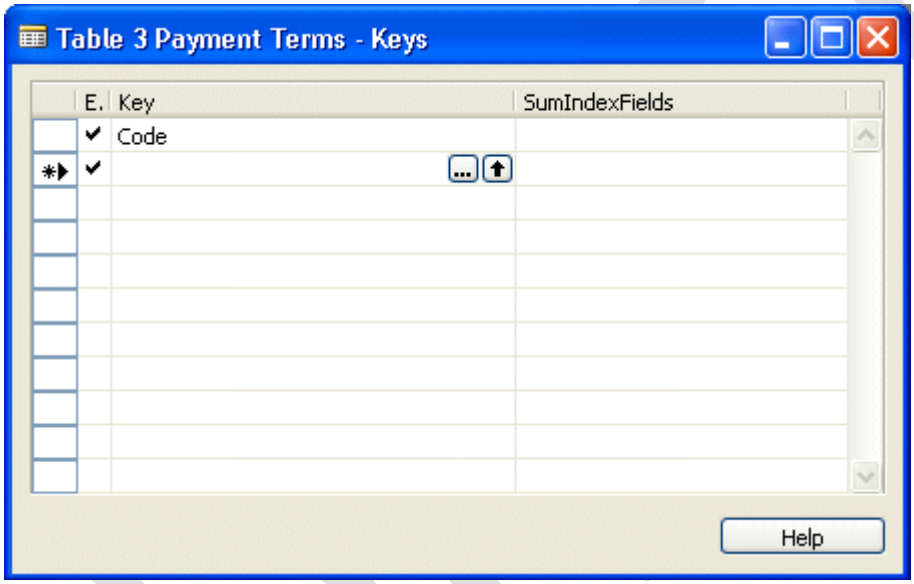
1. Щелкните Сервис, Object Designer откроется окно ***Object Designer,*** щелкните кнопку **Table**, чтобы увидеть список таблиц.

2. Выберите таблицу и щелкните кнопку **Design**. C/SIDE отобразит таблицу в окне **Table Designer**.

DRAFT

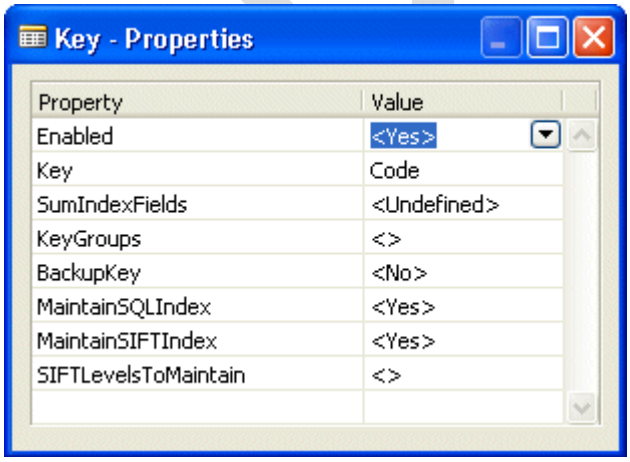


3. Щелкните Вид, Keys:



4. Установите курсор в строке, где определен ключ, для которого вы хотите получить доступ к свойствам.

5. Щелкните Вид, Properties. C/SIDE отобразит окно **Properties**:



6. Если вы хотите изменить значение свойства, просто введите новое значение в окне **Properties**. После ввода нового значения нажмите клавишу ENTER или просто уберите курсор с поля, чтобы обновить свойство.

7. Для получения Справки по свойству укажите на него в окне Properties и нажмите клавишу F1.

6.4 Идентификаторы, Типы Данных, Форматы Данных в SQL Server для Dynamics NAV

В данном разделе будет описаны идентификаторы, типы данных и форматы данных, используемые в SQL Server.

Наименование Идентификаторов

Идентификаторы таблиц и столбцов в SQL Server основываются на наименованиях соответствующих таблиц и полей в Определении Таблиц Dynamics NAV. Если в свойстве таблицы *DataPerCompany* установить значение Да, то перед наименованием таблицы в SQL Server будет добавлен префикс в виде наименования фирмы. Два наименования разделены символом ($). Например, название таблицы ***Customer*** для CRONUS Россия ЗАО на SQL Server будет выглядеть как CRONUS Россия ЗАО$Customer. Если в свойстве *DataPerCompany* установлено значение Нет, то префикс не используется.

Первичный ключ таблицы Dynamics NAV на SQL Server создается как ограничение первичного ключа. Наименование первичного ключа основывается на наименовании таблицы с суффиксом *$0*, например, CRONUS Россия ЗАО$Customer$0. Все остальные ключи таблицы Dynamics NAV, которые должны создаваться и поддерживаться в SQL Server, т.е. для которых в свойстве *MaintainSQLIndex* установлено значение Да, создаются как индексы. Их наименование состоит из префикса $ и внутреннего идентификатора ключа. Примеры: *$1* и *$4*.

Если для базы данных включена поддержка видов (maintain SQL views) для идентификаторов языка, система создаст представление данных SQL (SQL view), добавив префикс идентификатора языка Windows (Windows language ID). Например, если требуется обратиться к фирме CRONUS Россия ЗАО на русском, то SQL view будет таким RUS$CRONUS Россия ЗАО$Клиент. Дополнительную информацию о многоязыковой функциональности см. ГЛАВА 23 МНОГОЯЗЫКОВАЯ ФУНКЦИОНАЛЬНОСТЬ.

Если для базы данных включена поддержка связей (maintain relationships), система создает ограничения внешнего ключа, используя наименование таблицы SQL Server и данные свойства *TableRelation*. Имя ограничения создается с использованием следующего формата:

<table name>FKT<referencing table ID>_F<referencing field ID>$T<referenced table ID>.

Далее приведен пример использования таблицы ***Customer***:

CRONUS Россия ЗАО$Customer$FK$T18_F107$T30

Это означает: *Таблица 18(Customer) поле 107 (No. Series) ссылается на таблицу 308 (No. Series).*

Создание в Dynamics NAV таблиц с ключами, содержащими SumIndexFields, приводит к созданию дополнительных представлений данных SQL, которые должны поддерживать функциональность SIFT. Эти представления данных SQL наименовываются на основании наименовании таблицы SQL Server, строки $VSIFT и внутреннего идентификатора ключа. Например, для ключа (G/L Account No.,Posting Date) таблицы ***G/L Entry*** (Фин. Книга Операций) представление данных будет следующим: CRONUS Россия ЗАО$G_L Entry$VSIFT$1.

Представление Типов Данных Dynamics NAV

Каждый доступный в Dynamics NAV тип данных связывается с соответствующим типом данных SQL Server в таблицах SQL Server для Dynamics NAV. Следующая таблица показывает, как соотносятся типы данных SQL Server с типами данных Dynamics NAV:

Тип Данных Dynamics NAV	Тип Данных SQL Server
Integer	INTEGER
Option	INTEGER
Code(n)	VARCHAR(n) INTEGER SQL_VARIANT
Text(n)	VARCHAR(n)
Decimal	DECIMAL(38,20)
Date	DATETIME
Time	DATETIME
DateTime	DATETIME
Boolean	TINYINT
Binary(n)	VARBINARY(n)
BLOB	IMAGE
DateFormula	VARCHAR(32)
TableFilter	VARBINARY(252)
BigInteger	BIGINT
Duration	BIGINT
GUID	UNIQUEIDENTIFIER
RecordID	VARBINARY(n)

Каждый тип данных SQL Server создан как NOT NULL, за исключением типа IMAGE, для которого разрешено пустое значение (NULL).

Совместимость Типов Данных

Некоторые типы данных SQL Server, упомянутые выше, совместимы с другими типами данных Dynamics NAV. Следующая таблица отображает расширенную совместимость типов данных SQL Server и типами данных Dynamics NAV:

Тип Данных SQL Server	Тип Данных Dynamics NAV
CHAR(n)	Code(n) Text(n) DateFormula
NCHAR(n)	Text(n)
NVARCHAR(n)	Text(n)
INTEGER	Code
TINYINT	Integer Option
SMALLINT	Integer Option

DRAFT

Тип Данных SQL Server	Тип Данных Dynamics NAV
NUMERIC(p,s), MONEY, SMALLMONEY, REAL, FLOAT(n), DECIMAL	Decimal Integer Option Boolean
SMALLDATETIME	Date
BIT	Integer Option Boolean
BINARY(n)	Binary(n)
TEXT	BLOB
NTEXT	BLOB
UNIQUEIDENTIFIER	Binary(16) Text(36)

Форматы Данных

При использовании SQL Server для Dynamics NAV, следует знать о влиянии формата данных на сравнение и сортировку данных.

Поля с Типом Данных Code

В SQL Server для Dynamics NAV, поля с типом данных code могут быть представлены несколькими типами данных SQL Server.

Для полей с типом данных Code есть свойство *SQL Data Type*, которое определяет, будет ли поле содержать целые числа, текстовые строки или их смесь. Данное свойство устанавливается следующим образом:

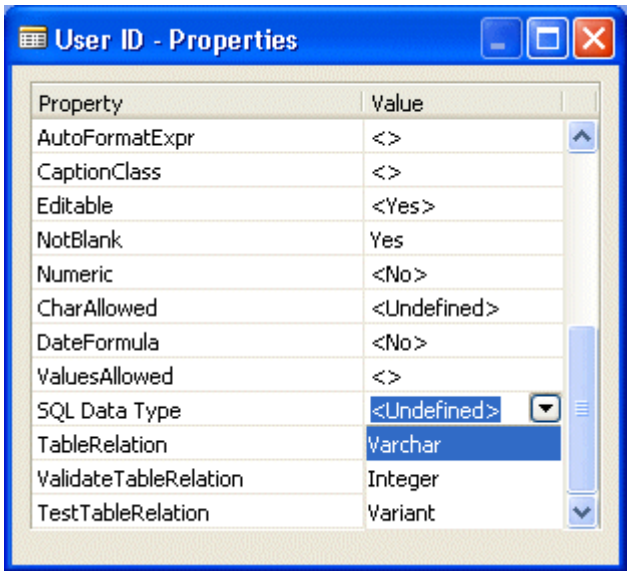
1. Щелкните Сервис, Object Designer.

2. Щелкните кнопку **Table** и выберите подходящую таблицу.

3. Щелкните кнопку **Design**.

4. Выберите поле, для которого тип данных указан как Code, затем щелкните Вид, Properties. Появится окно ***Properties***:

DRAFT



Свойство *SQL Data Type* может принимать следующие значения *Varchar*, *Integer* или *Variant*. Если оставить значение *Undefined,* это аналогично выбору значения *Varchar,* которое является значением по умолчанию. При создании таблицы на SQL Server для Dynamics NAV, значения поля с типом данных Code хранятся в колонках VARCHAR, INTEGER или SQL_VARIANT в соответствии со значениями свойства *SQL Data Type - Varchar*, *Integer* или *Variant*.

При установке в свойство *SQL Data Type* значение *Varchar*, все значения в данном поле сравниваются и сортируются как символьные данные, в том числе и числовые значения.

Если установить в свойстве *SQL Data Type* значение *Integer*:

- Все значения в данном поле сравниваются и сортируются как целые. Буквенно-числовые значения не могут храниться в данном поле.
- Если в данное поле с помощью внешних инструментов записать отрицательные значения, Dynamics NAV не сможет их прочесть.
- Значение "0"(ноль) используется для представления пустой строки в Dynamics NAV.
- Не числовые значения или любые числовые значения, начинающиеся с "0"(ноль), не могут быть введены в данное поле.

Если установить в свойстве *SQL Data Type* значение *Variant*:

- Значения в данном поле хранятся и сортируются также как их базовые типы. Числовые значения при сортировке оказываются после буквенно-числовых.
- Данные введенные в данное поле в Dynamics NAV хранятся либо как VARCHAR, либо как INTEGER, в зависимости от введенного значения.
- Любые значение начинающиеся с "0"(ноль) могут быть введены в данное поле.

  Примечание

  Не все инструменты сторонних производителей, которые используются для доступа к данным в базе данных SQL Server, поддерживают тип данных *Variant*.

Поля с Типом Данных Date и Time

SQL Server хранит информацию об обоих типах данных Date и Time в колонках с типом DATETIME и SMALLDATETIME. Для полей Date, Dynamics NAV использует только часть поля, а в качестве времени помещает константы. Для нормальных дат это 00:00:00:000. Для закрывающей даты это 23:59:59:000 для DATETIME и 23:59:00:000 для SMALLDATETIME.

Неопределенная дата Dynamics NAV в SQL Server отображается как самая ранняя доступная дата: 01-01-1753 00:00:00:000 для DATETIME, и 01-01-1900 00:00:00:000 для SMALLDATETIME.

DRAFT

Для полей с типом данных Time может использоваться только тип данных SQL Server DATETIME. Dynamics NAV используется только часть поля для хранения времени, а в качестве данных использует константу: 01-01-1754. Неопределенное время в Dynamics NAV представляется таким же образом как неопределенная дата.

Чтобы Dynamics NAV корректно интерпретировал значения даты и времени, указанные выше форматы должны использоваться также и во внешних таблицах и представления данных подключенных к Dynamics NAV. Дополнительную информацию об этом см. в 7.4 Связанные Объекты (Linked Objects).

Чтобы преобразовать формат колонок DATETIME или SMALLDATETIME, так чтобы они могли использоваться как поля с типом данных Date, можно использовать оператор UPDATE к данным. Дальше приведены примеры выражений:

UPDATE [My Table] SET [My Date] = CONVERT(CHAR(10), [My Date], 102)

Для закрывающей даты, может использоваться стиль 120 для функции CONVERT. Чтобы преобразовать поле с типом данных Time используется выражение:

UPDATE [My Table] SET [My Time] = CAST('1754-01-01 '+CONVERT(CHAR(8), [My Time], 108) AS DATETIME)

Как альтернативу изменению данных, можно создать представление данных, для которого применяются необходимые преобразования. Однако вы не сможете обновлять представление данных созданные таким образом, поэтому более эффективно выполнить конвертацию для всех строк.

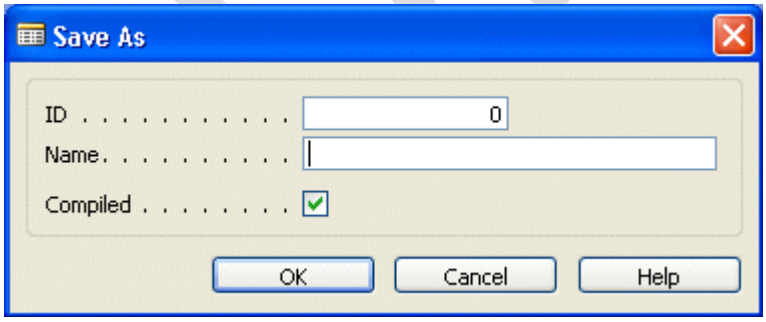
> **Примечание**
>
> Информация в данном разделе относится к типам данных *Date* и *Time,* и не применяется для полей с типом данных *DateTime*.

6.5 Сохранение Таблиц и Просмотр Отсортированных Данных

После проектирования полей и ключей для новой таблицы, перед тем как таблицу можно будет использовать ее необходимо сохранить в базе данных. После сохранения таблица появится в списке таблиц, отображаемом в окне Object Designer.

Чтобы сохранить таблицу в базе данных:

1. Убедитесь, что фокус установлен в Table Designer на нужной таблице. Затем щелкните Файл, Save. C/SIDE покажет следующее окно:



2. В поле **ID** введите номер, который будет использоваться как уникальный идентификатор таблицы. Существует ряд ограничений на использование номеров. Для дополнительной информации обратитесь к сертифицированному партнеру Microsoft с компетенцией Business Solutions.

Просмотр Данных

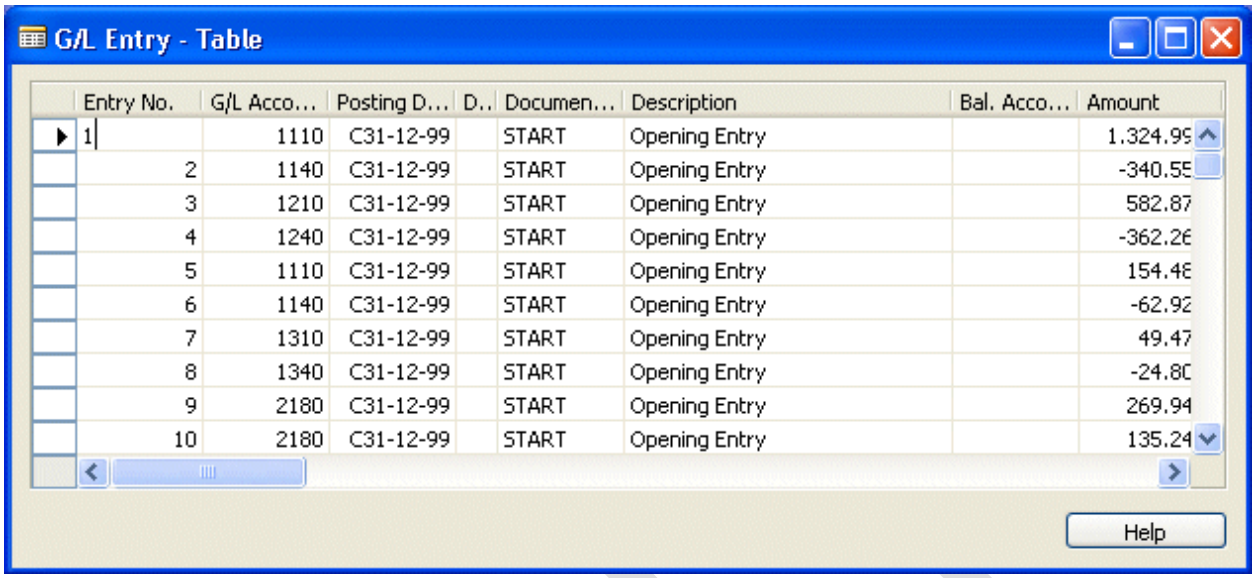
Обычно для просмотра данных в таблицах используются формы, но также данные можно посмотреть, открыв таблицу непосредственно из Object Designer.

DRAFT

Чтобы посмотреть данные в таблице, не используя форму:

1. Щелкните Object Designer (SHIFT+F12) и выберите таблицу.

2. Щелкните кнопку Run, C/SIDE отобразит данные в табличном виде:



Entry No.	G/L Acco...	Posting D...	D..	Documen...	Description	Bal. Acco...	Amount
1	1110	C31-12-99		START	Opening Entry		1.324.99
2	1140	C31-12-99		START	Opening Entry		-340.55
3	1210	C31-12-99		START	Opening Entry		582.87
4	1240	C31-12-99		START	Opening Entry		-362.26
5	1110	C31-12-99		START	Opening Entry		154.48
6	1140	C31-12-99		START	Opening Entry		-62.92
7	1310	C31-12-99		START	Opening Entry		49.47
8	1340	C31-12-99		START	Opening Entry		-24.80
9	2180	C31-12-99		START	Opening Entry		269.94
10	2180	C31-12-99		START	Opening Entry		135.24

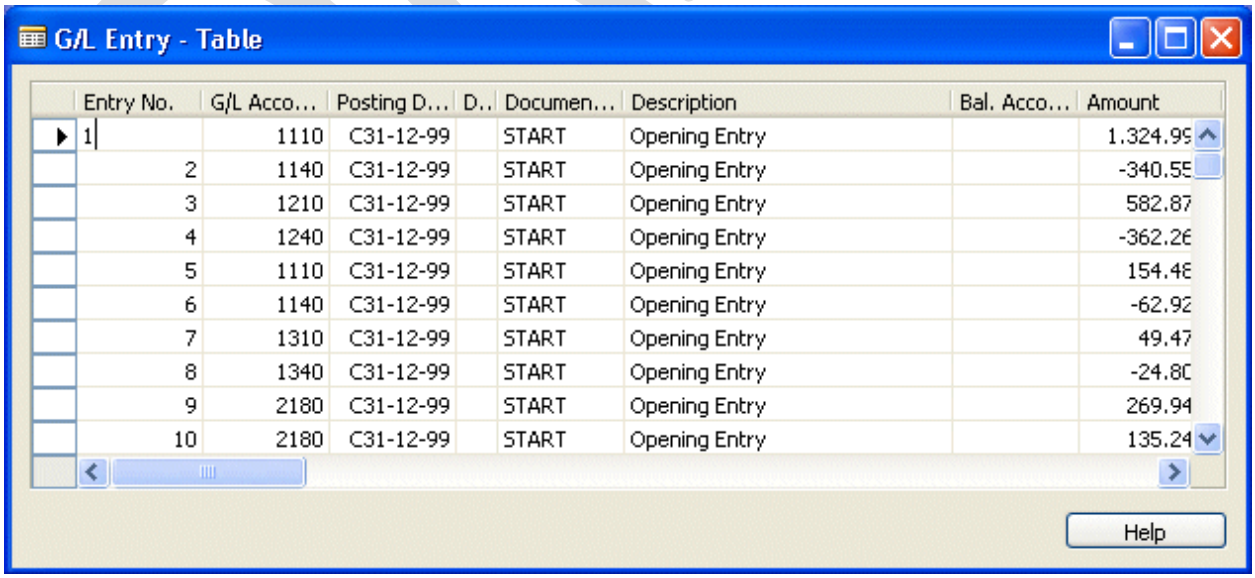
Порядок Сортировки

Порядок, в котором информация выводится в окно, определяется сортировкой согласно текущему ключу. Если для таблицы существует несколько ключей, можно переключать порядок сортировки согласно существующим ключам.

Изменение Порядка Сортировка

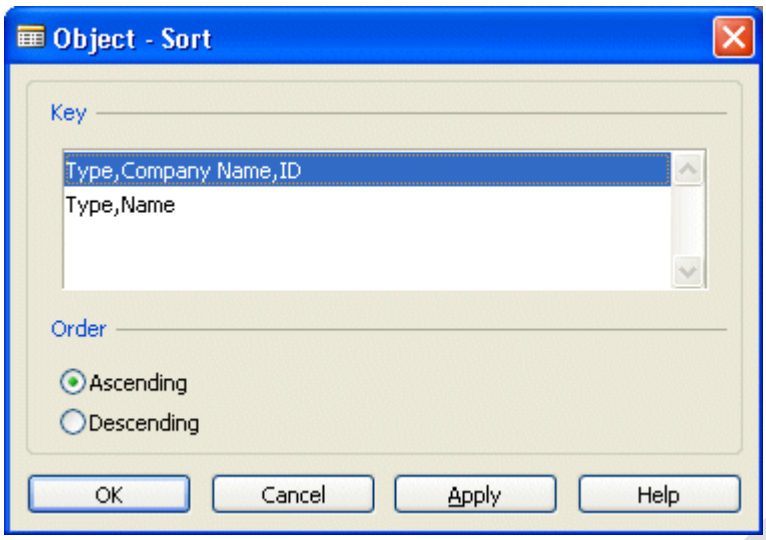
Чтобы посмотреть данные с различными порядками сортировки:

1. Откройте Object Designer (SHIFT+F12), выберите таблицу, с которой собираетесь работать и щелкните кнопку Run, чтобы открыть таблицу.



Entry No.	G/L Acco...	Posting D...	D..	Documen...	Description	Bal. Acco...	Amount
1	1110	C31-12-99		START	Opening Entry		1.324.99
2	1140	C31-12-99		START	Opening Entry		-340.55
3	1210	C31-12-99		START	Opening Entry		582.87
4	1240	C31-12-99		START	Opening Entry		-362.26
5	1110	C31-12-99		START	Opening Entry		154.48
6	1140	C31-12-99		START	Opening Entry		-62.92
7	1310	C31-12-99		START	Opening Entry		49.47
8	1340	C31-12-99		START	Opening Entry		-24.80
9	2180	C31-12-99		START	Opening Entry		269.94
10	2180	C31-12-99		START	Opening Entry		135.24

2. Щелкните Вид, Сортировка (SHIFT+F8), C/SIDE покажет следующее окно:



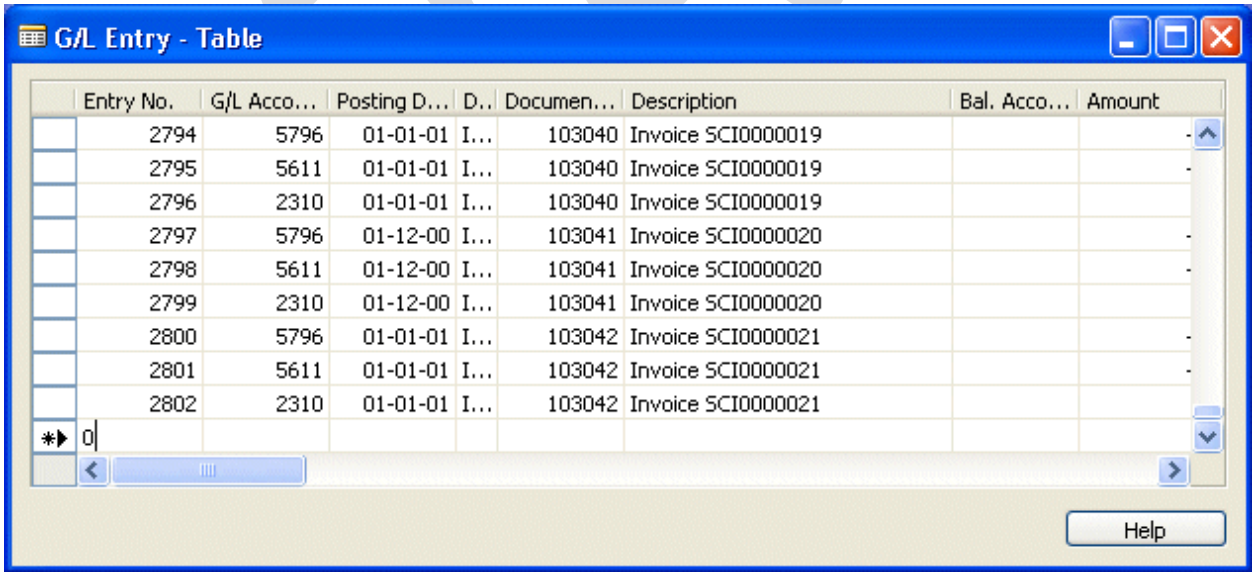
3. Выберите ключ, определяющий подходящий порядок сортировки и укажите, как должны показываться записи: по возрастанию или по убыванию. Щелкните кнопку OK или Применить, чтобы применить новый ключ. Если щелкнуть кнопку OK, то окно **Сортировать** закроется. А если щелкнуть кнопку Применить, то окно **Сортировать** останется открытым. Использование кнопки Применить удобно, если необходимо часто изменять порядок сортировки.

Добавление Записей без Использования Форм

Обычно для ввода данных в таблицы используются формы, но также данные можно ввести в таблицу непосредственно.

Чтобы добавить данные в таблицу, не используя формы, следует:

1. Открыть Object Designer (SHIFT+F12), выбрать таблицу и щелкнуть кнопку Run. C/SIDE отобразит данные в табличном виде:



Entry No.	G/L Acco...	Posting D...	D..	Documen...	Description	Bal. Acco...	Amount
2794	5796	01-01-01	I...	103040	Invoice SCI0000019		-
2795	5611	01-01-01	I...	103040	Invoice SCI0000019		-
2796	2310	01-01-01	I...	103040	Invoice SCI0000019		
2797	5796	01-12-00	I...	103041	Invoice SCI0000020		-
2798	5611	01-12-00	I...	103041	Invoice SCI0000020		-
2799	2310	01-12-00	I...	103041	Invoice SCI0000020		
2800	5796	01-01-01	I...	103042	Invoice SCI0000021		-
2801	5611	01-01-01	I...	103042	Invoice SCI0000021		-
2802	2310	01-01-01	I...	103042	Invoice SCI0000021		
0							

2. Поместите курсор в пустую строку или нажмите клавишу F3 для создания пустой строки. Введите данные в поля и нажмите клавишу ENTER. Для перемещения по полям можно использовать TAB, SHIFT+TAB и клавиши управления курсором.

3. После завершения ввода данных закройте таблицу (данные не нужно сохранять т.к. записи сохраняются и обновляются после того как курсор покидает поле, в которое перед этим вносились данные).

6.6 Специальные Поля Таблиц

В дополнение к обычным полям, которые применяются для хранения данных в таблицах, есть еще три вида специальных полей которые используются для получения данных:

- SumIndexFields (Суммирующие Поля)
- FlowFields (Вычисляемые Поля)
- FlowFilter

Что Такое Суммирующие Поля SumIndexFields?

SumIndexField – это десятичное поле, которое можно присоединить к определению ключа. Это фундаментальная особенность баз данных в формате Dynamics NAV, составляющая основу для FlowFields. Суммирующие поля SumIndexFields позволяют быстро выполнять вычисление сумм числовых столбцов в таблицах, в том числе даже в таблицах, содержащих тысячи записей. Это достигается благодаря тому, что поля SumIndexFields обновляются при обновлении записей базы данных.

Какие Преимущества Дают Поля SumIndexFields?

SumIndexField позволяет быстро вычислять сумму столбцов и отображать ее через FlowFields.

Например, необходимо получить сумму всех значений в поле **Amount**. В обычной системе баз данных СУБД приходится получать доступ к каждой записи и добавлять каждое значение поля **Amount.** Это занимает много времени, если база данных содержит тысячи записей. Благодаря системе Dynamics NAV для получения суммы поля Amount этих записей доступ должен быть получен всего лишь дважды (если используется наиболее подходящий ключ).

Эта специальная индексная структура, SumIndexField, связана с ключом. Каждый ключ может содержать максимум 20 полей SumIndexFields.

При проектировании базы данных десятичное поле может быть связано с ключом как SumIndexField. Оно инструктирует СУБД создать и обновить структуру, содержащую накопленную сумму значений в столбце. При выборе нового текущего ключа становится доступным любое поле SumIndexFields, связанное с ним.

Что Такое Вычисляемые Поля FlowFields?

Поле FlowFields – мощный инструмент системы баз данных C/SIDE. Вычисляемые поля FlowField – фундаментальная концепция, которая оказывает значительное влияние на проектирование приложения C/SIDE.

Поле FlowField и лежащая в его основе концепция суммирующих полей SumIndexFields разработаны для увеличения производительности в таких областях, как вычисление баланса клиентов. В традиционных системах баз данных это предполагает необходимость многократного получения доступа и выполнения множества вычислений, прежде чем будет получен результат. Как такой результат будет получен немедленно при использовании FlowFields, станет ясно при прочтении остальной части этого раздела.

Поле FlowFields не является постоянной частью данных таблицы. Вычисляемое поле можно представить себе в качестве виртуального поля, являющегося расширением данных таблицы. Фактически информация, содержащаяся в динамическом поле, существует только во время выполнения. Значения в вычисляемом поле автоматически инициализируются в 0 (нуль). Для обновления динамического поля необходимо использовать C/AL-функцию <Record>.CALCFIELDS. Обратите внимание, что, когда динамическое поле является прямым исходным выражением элемента управления формы, оно рассчитывается автоматически в момент отображения формы.

DRAFT

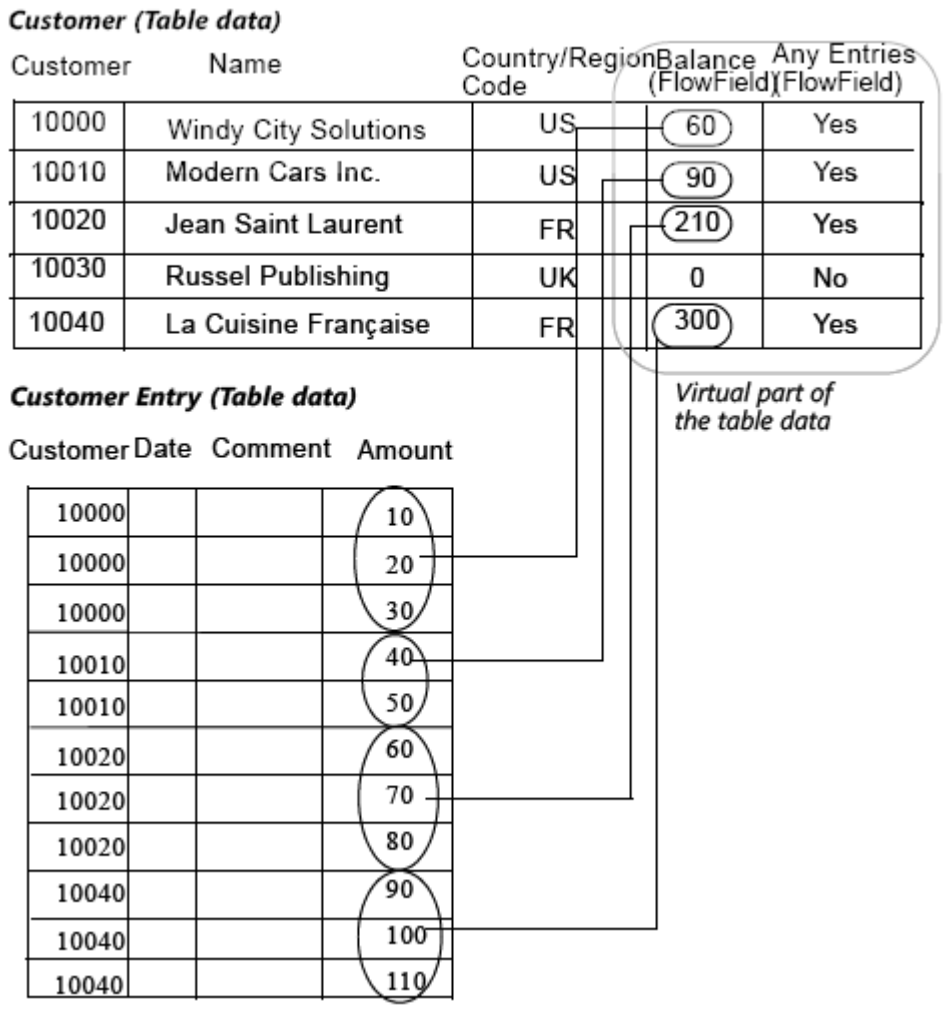
Типы Вычисляемых Полей (FlowField)

Существует семь типов динамических полей:

Тип FlowField	Тип Данных Поля	Описание
Sum	decimal	Сумма значений указанного набора записей в столбце таблицы
Average	decimal	Среднее значение указанного набора записей в столбце таблицы.
Exist	boolean	Показывает, существуют ли какие-либо записи в указанном наборе таблицы.
Count	integer	Число записей в указанном наборе таблицы.
Min	любой	Минимальное значение в столбце в указанном наборе таблицы.
Max	любой	Максимальное значение в столбце в указанном наборе таблицы.
Lookup	любой	Производится поиск в столбце другой таблицы.

Пример

Рассмотрим таблицу ***Customer*** представленную на следующем рисунке. Эта таблица содержит два вычисляемых поля. Поле **Any Entries –** вычисляемое поле типа Exist, поле **Balance –** вычисляемое поле типа Sum.



На рисунке показано, что вычисляемое поле **Balance** для клиента 10000 является итогом по полю **amount** операций, связанных с клиентом 10000

Sum = 10 + 20 + 30 = 60.

Значения, отображаемые в полях **Balance** таблицы ***Customer*** для клиентов 10010, 10020, 10040 рассчитываются таким же образом. Для клиента 10030 значение 0 (ноль), так как в таблице ***Customer Entry*** нет операций для клиента 10030.

Формула расчета для поля **Balance**:

Sum("Customer Entries".Amount WHERE(CustNo=FIELD(CustNo)))

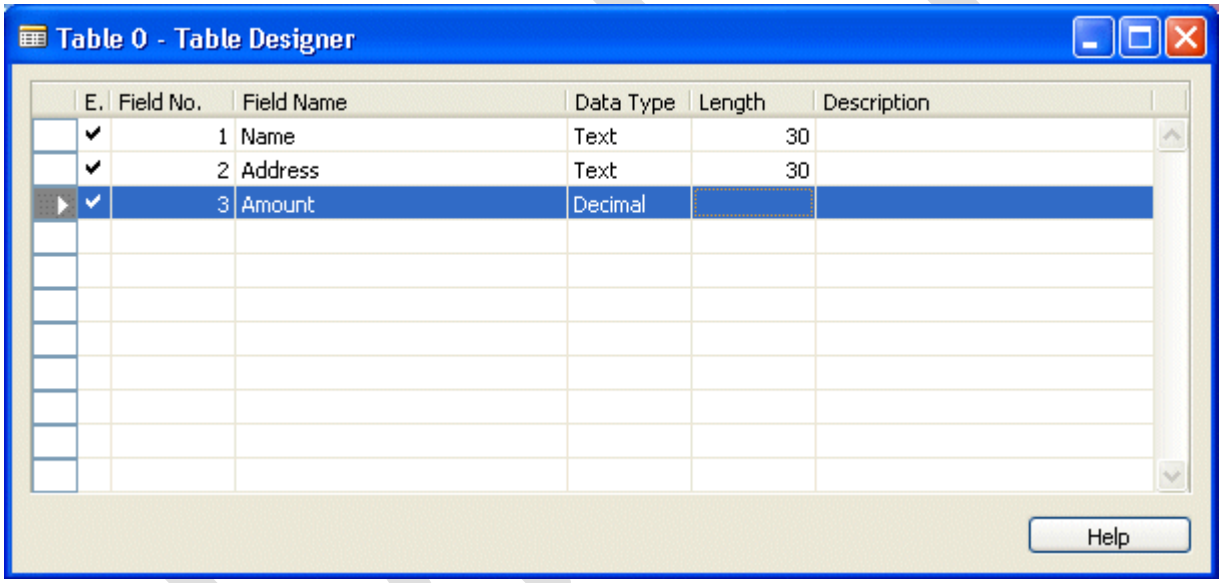
Соответственно для поля **Any Entries**, которое показывает, существуют ли операции для данного клиента, формула расчета будет такой:

Exist("Customer Entries" WHERE(CustNo=FIELD(CustNo)))

Создание Вычисляемого Поля

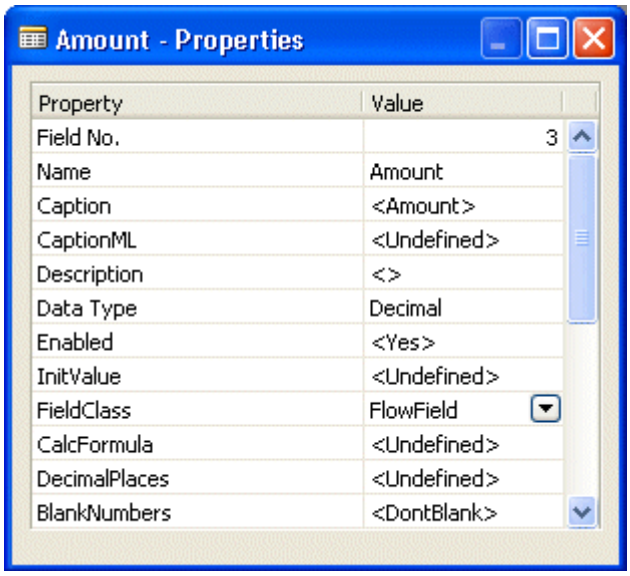
Чтобы создать вычисляемое поле:

1. Откройте Object Designer (SHIFT+F12), затем откройте таблицу, к которой требуется добавить вычисляемое поле в Table Designer:



2. Щелкните по строке, которая будет определена как вычисляемое поле, в данном случае это поле **Amount**.

3. Щелкните Вид, Properties (SHIFT+F4), будет открыто окно ***Properties*** для поля **Amount**:



4. Измените значение свойства *FieldClass* с Normal на FlowField.

5. Введите формулу расчета вычисляемого поля. Это делается в свойстве *CalcFormula*. Далее будет рассказано, как это сделать.

Формулы Расчета Вычисляемого Поля и Свойство *CalcFormula*

Поле FlowField всегда связывается с формулой для производства вычислений, которая определяет, как рассчитывается значение в динамическом поле. Ниже приведен синтаксис свойства CalcFormula:

<CalculationFormula> ::=

[-]Exist(<TableNo> [WHERE (<TableFilters>)]) |

Count(<TableNo> [WHERE (<TableFilters>)]) |

[-]Sum(<TableNo>.<FieldNo> [WHERE(<TableFilters>)])|

[-]Average(<TableNo>.<FieldNo> [WHERE (<TableFilters>)]) |

Min(<TableNo>.<FieldNo> [WHERE (<TableFilters>)]) |

Max(<TableNo>.<FieldNo> [WHERE (<TableFilters>)]) |

Lookup(<TableNo>.<FieldNo> [WHERE (<TableFilters>)])

<TableFilters> ::=

[<TableFilter> {,<TableFilter>}]

<TableFilter> ::=

<DstFieldNo>=CONST(<FieldConst>) |

<DstFieldNo>=FILTER(<Filter>) |

<DstFieldNo>=FIELD(<SrcFieldNo>) |

<DstFieldNo>=FIELD(UPPERLIMIT(<SrcFieldNo>)) |

<DstFieldNo>=FIELD(FILTER(<SrcFieldNo>)) |

<DstFieldNo>=FIELD(UPPERLIMIT(FILTER(<SrcFieldNo>)))

Где:

Символ	Описание

DRAFT

Символ	Описание
<TableNo>	Определяется таблица, содержащая информацию, используемую в Вычисляемом Поле.
<FieldNo>	Указывается столбец, на основании которого требуется рассчитать значение
<TableFilters>	Список фильтров, используемых при расчете вычисляемого поля
<TableFilter>	Фильтр таблицы могут быть: • константа, • выражение-фильтр, • фильтр на основании значения обычного поля или поля FlowFilter (поля FlowFilter будут рассмотрены в следующем разделе). Обратите внимание, что для второй таблицы должен существовать ключ, содержащий все поля, используемые при установке фильтров.
<DstFieldNo>	Фильтруемое поле (поле на которое накладывается фильтр).
<SrcFieldNo>	Поле-источник фильтра.
<Filter>	Выражение-фильтр, например: 10|20..30.

Создание, Просмотр и Редактирование Формулы Расчета Вычисляемого Поля

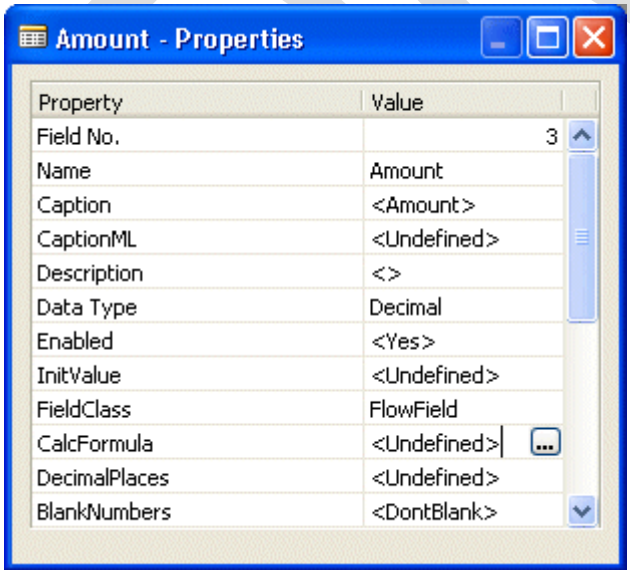
Чтобы создать, просмотреть или отредактировать формулу расчета вычисляемого поля надо:

1. Открыть Object Designer (SHIFT+F12), затем открыть требуемую таблицу.

2. Выбрать поле, для которого будет создаваться или редактироваться формула расчета вычисляемого поля.

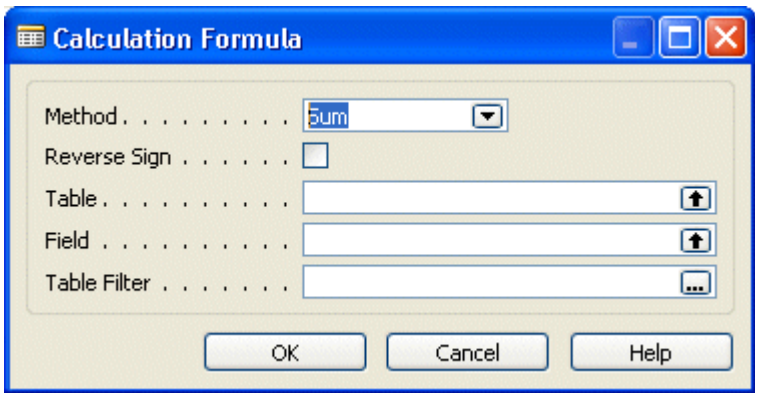
3. Щелкнуть Вид, Properties (SHIFT+F4) чтобы открыть окно ***Properties***.

4. Найти свойство *CalcFormula*:



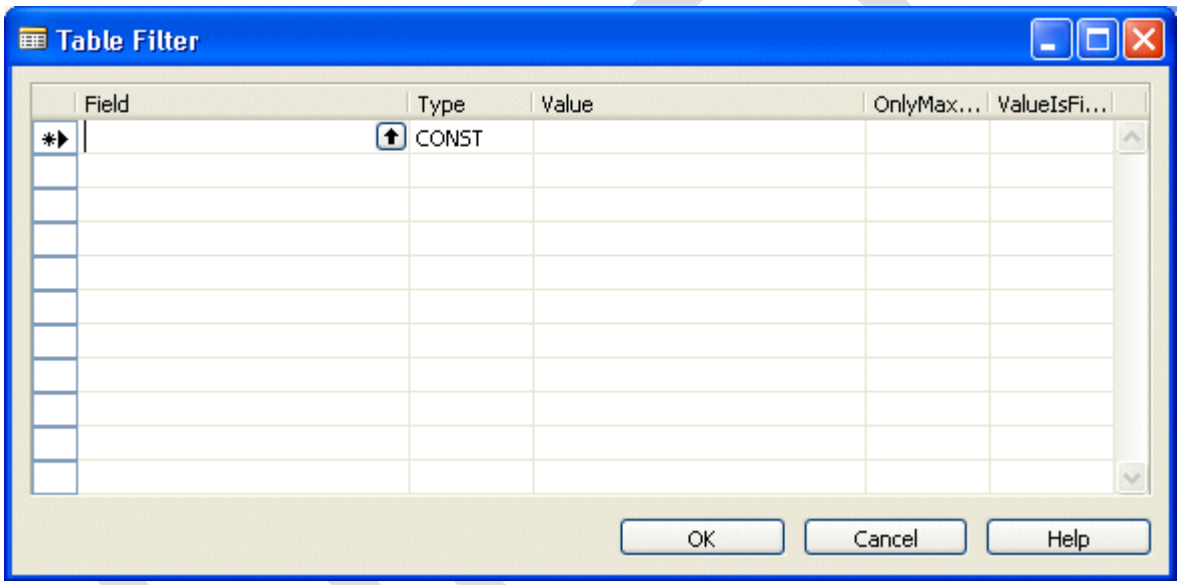
5. Формулу расчета можно ввести непосредственно в поле, также можно щелкнуть кнопку AssistButton.

Если щелкнуть по кнопке AssistButton откроется следующее окно:



6. Выберите подходящий метод расчета из выпадающего списка. Для методов расчета Sum и Average можно указать, что система должна выводить результат с обратным знаком, для этого нужно установить флаг в поле Reverse Sign. Выберите таблицу и столбец (поле), из которых система будет получать информацию. При необходимости можно добавить фильтры таблиц, чтобы указать определенный набор записей.

Щелкните кнопку AssistButton в правой части поля Table Filter, откроется окно ***Table Filter***:

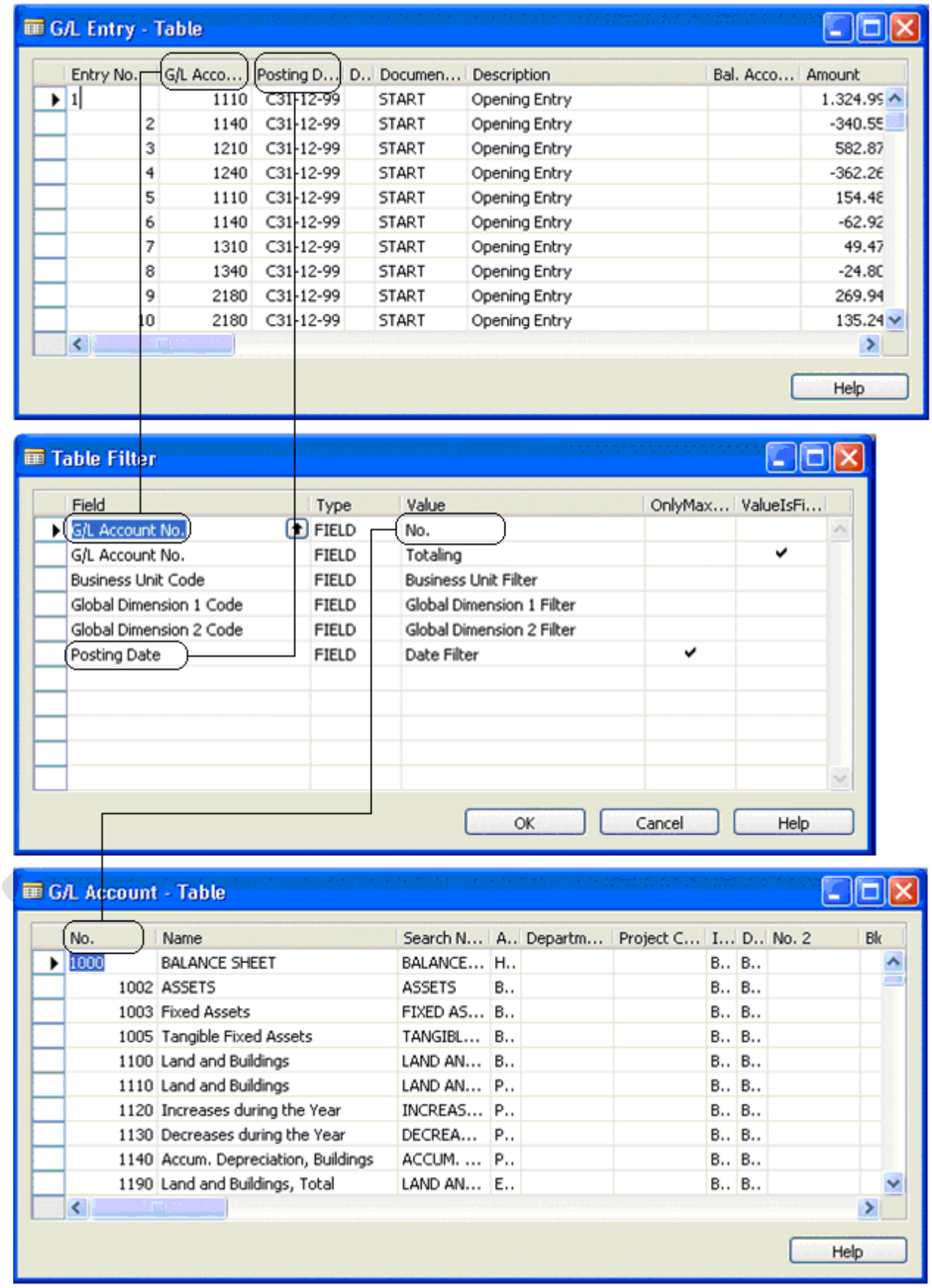


7. В каждой строке данного окна можно указать фильтр для поля, при этом требуется указать Field (Поле), Type (Тип фильтра) и Value (Значение). Также доступны опции OnlyMaxLimit и ValueIsFilter. В следующем примере будет показано, откуда берутся данные в данном окне.

Пример

Поле **Balance at Date** таблицы ***G/L Account (Фин. Счет)*** – это вычисляемое поле с типом данных Decimal. Это поле рассчитывается на основании значения столбца **Amount** в таблице ***G/L Entry***.

Поле **Amount** содержит данные, которые должны быть просуммированы. Это поле определено как SumIndexField в ключе таблицы ***G/L Entry***.

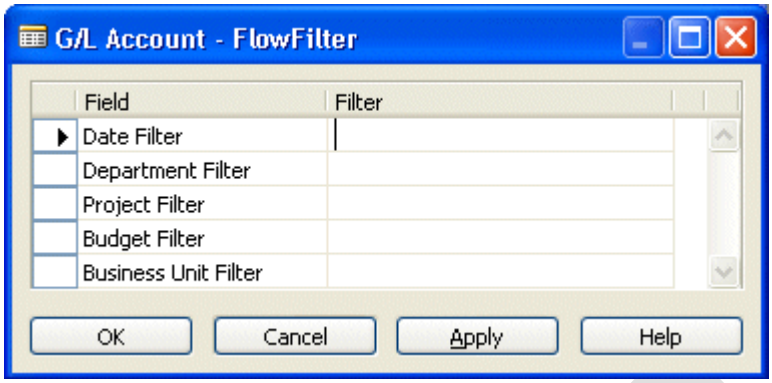


Поле **Field** в окне Table Filter содержит ссылку на поле (столбце) в таблице ***G/L Entry***.

Столбец **Value** окна Table Filter содержит ссылку на поле (столбец) в таблице ***G/L Account***.

Часть полей таблицы ***G/L Account*** являются FlowFilter. При вводе в данные поля выражений-фильтров, пользователь может влиять на расчет вычисляемого поля (например, **Balance at Date**).

Пользователь вводит значения фильтров в окне FlowFilter:



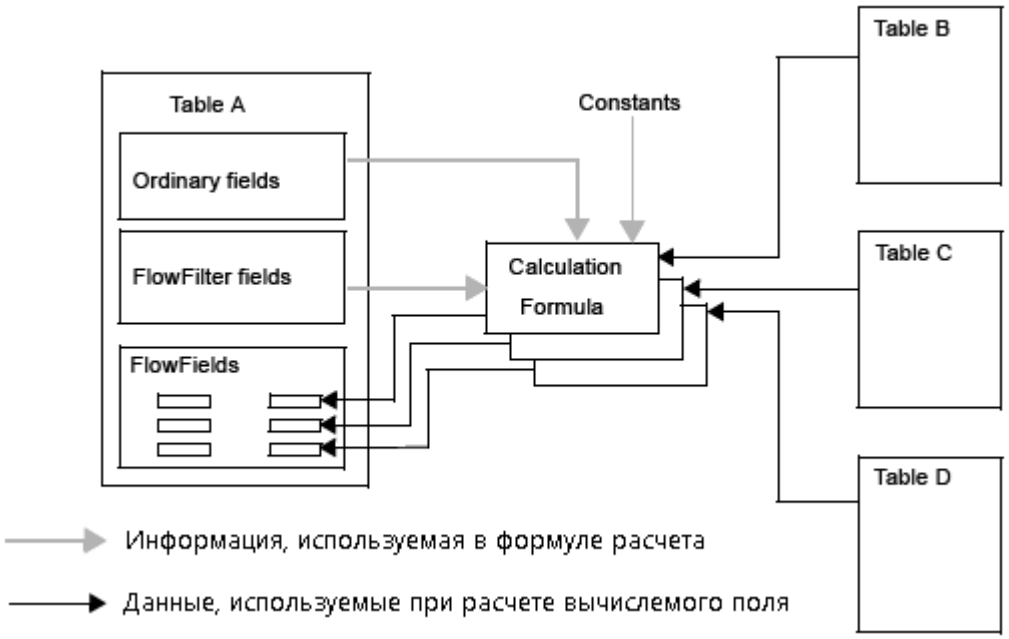
Если в поле **Date Filter** ввести выражение-фильтр для дат, оно будет применено к полю **Date** в таблице ***G/L Entry***.

Поле OnlyMaxLimit используется, чтобы убрать нижнюю границу диапазона, определенную выражением-фильтром. Например, если выражение-фильтр определено как диапазон x..y, установка флага в поле OnlyMaxLimit преобразует его в выражение..y.

Флаг в поле ValueIsFilter указывает, как система должна интерпретировать значение в поле указанном в столбце Value. Например, поле может содержать значение 1000..2000, установка флага в поле ValueIsFilter укажет системе, что она должна интерпретировать его как выражение-фильтр, а не как конкретное значение.

Использование Полей FlowFilter в Формулах Расчета Вычисляемого Поля

Пользователю может потребоваться ограничить расчет значения вычисляемого поля, так чтобы в него включались значения, удовлетворяющие определенным условиям. Например, пользователю может потребоваться сумма по полю **Amount** для операций клиента введенных в Апреле. Это возможно, если приложение было спроектировано, так чтобы при расчете вычисляемых полей (FlowFields) использовались FlowFilter.

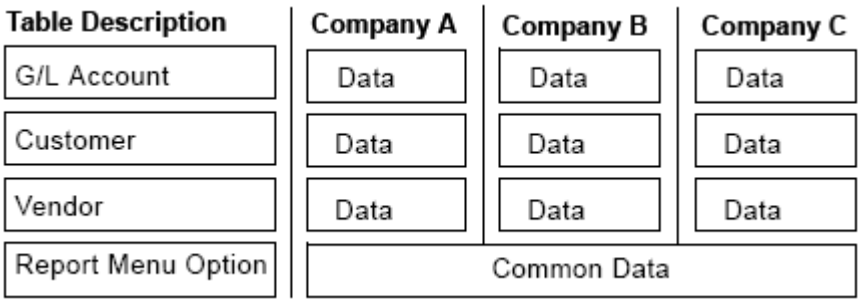


Данный рисунок иллюстрирует связь между двумя различными типами полей базы данных и формулой расчета. Фильтры, определенные в формуле расчета могут состоять из констант, значений из обычных полей и фильтров, задаваемых как параметры в полях FlowFilter. В полях FlowFilter пользователи могут вводить значения фильтров (используя обычный пользовательский интерфейс), и эти значения фильтров будут влиять на результаты расчета вычисляемых полей (FlowField).

6.7 Разделение Таблиц Между Фирмами

СУБД может одновременно работать только с одной логической базой данных, но база данных может быть разделена на одну и более фирм. Фирма это раздел базы данных, ее основная задача выделять и группировать данные в базе данных. Поля и таблицы идентифицируются по номерам, а фирма по ее наименованию. Фирма объединяет одну или несколько таблиц в одну логическую структуру, которая определяется наименованием фирмы. После открытия базы данных или подключения к серверу, следует открыть фирму. Как это делается описано в руководстве *Installation and System Management* для используемого вами варианта сервера.

Представим базу данных, включающую, четыре таблицы как показано на следующем рисунке.



Слева указанны четыре Описания Таблиц, они логически распределены между тремя фирмами. Записи таблиц ***G/L Account***, ***Customer*** и ***Vendor*** имеют одинаковую структуру и поля, хотя они принадлежат трем разным фирмам. Они различаются хранящимися в них данными. В базе данных не хранятся избыточные данные, так как информация из Описания Таблицы может использоваться для таблиц, принадлежащих разным фирмам. Это позволяет уменьшить размер базы данных.

Даже после выбора определенной фирмы, вы можете получить доступ к любой таблице, принадлежащей любой другой фирме. Для этого используется функция языка C/AL: <Record>.CHANGECOMPANY.

С ее помощью указывается другая фирма, к которой требуется получить доступ.

К одной таблице одной фирмы могут получать доступ одновременно несколько приложений. Как СУБД управляет множественными подключениями описано в разделе Что Такое Блокировка Таблиц?.

ГЛАВА 7 ИЗМЕНЕНИЕ И ПОДДЕРЖКА ТАБЛИЦ

После создания таблиц, возможно, потребуется использовать возможности предоставляемые свойствами и триггерами. Устанавливая свойства для таблиц, можно указать как должна вести себя таблица в определенных ситуациях. Добавляя C/AL код в триггеры, можно изменить поведение системы.

В данной главе будет показано, как использовать свойства и триггеры при проектировании таблиц. Затем будет показано, как устанавливать связи между таблицами. И наконец, в главе будет рассказано о проблемах, с которыми можно столкнуться при изменении таблиц, содержащих данные.

- Использование триггеров полей и таблиц
- Создание связей между таблицами
- Изменение таблиц, которые содержат данные
- Связанные объекты (Linked Objects)

7.1 Использование Триггеров Таблиц и Полей

Среда C/SIDE распознает некоторые ситуации при использовании таблиц, например, вставку или изменение данных. При этом систему можно обязать выполнять определенный C/AL код, указанный в соответствующих триггерах. Триггеры – это предопределенные функции, которые выполняются при наступлении некоторых событий. Изначально это функции пусты. Добавление C/AL кода в триггеры позволяет изменять поведение системы.

Триггеры таблиц C/SIDE можно разделить на две категории:

- Триггеры таблиц
- Триггеры полей

К триггерам таблиц C/SIDE относятся:

Название Триггера Таблиц	Когда выполняется
OnInsert	При вставке новой записи в таблицу
OnModify	При изменении существующей записи таблицы
OnDelete	При удалении записи из таблицы
OnRename	Когда изменяется поле, входящее в первичный ключ

Поля таблиц имеют следующие триггеры:

Название Триггера Поля	Когда выполняется
OnValidate	При вводе данных в поле или при вызове из C/AL кода функции <Record>.VALIDATE
OnLookup	При нажатии на кнопку Выбор (F6)

Если вы не знакомы с программированием на языке C/AL, ознакомьтесь с ГЛАВА 16 ВВЕДЕНИЕ В ЯЗЫК C/AL.

Для создания или изменения триггера таблицы или поля:

DRAFT

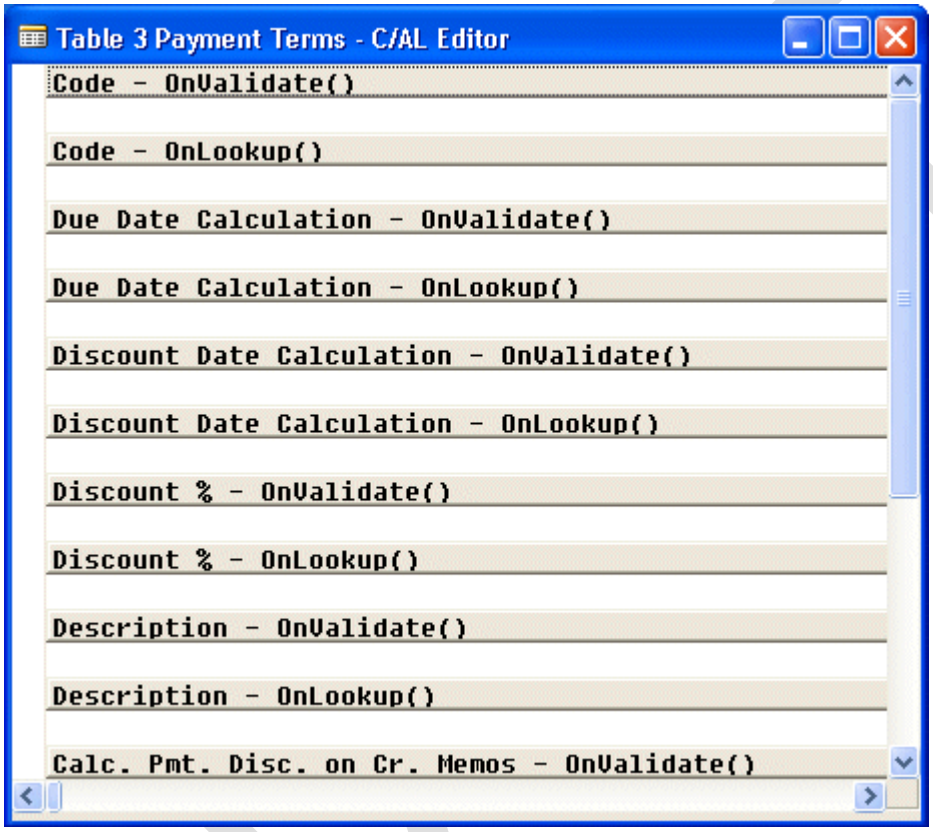
1. Откройте Object Designer (SHIFT+F12), затем щелкните кнопку **Table**, чтобы открыть список таблиц.

2. Выберите нужную таблицу и щелкните кнопку **Design**. Система откроет **Table Designer**, в котором перечислены все поля таблицы.

3. Щелкните Вид, C/AL Code (F9).

4. Система определит, какой код следует отображать, основываясь на положении курсора в окне Table Designer. Если курсор установлен в каком-то определенном поле, окно C/AL Editor автоматически прокрутит код так, чтобы первый триггер, связанный с данным полем, отображался вверху экрана. Если курсор установлен в пустой строке, система покажет первый триггер, связанный непосредственно с таблицей.

Положение курсора в окне Table Designer не ограничивает доступ к другим триггерам. Вы можете прокручивать код вверх и вниз, чтобы получить доступ к нужным триггерам в C/AL editor.



5. Теперь можно добавить или изменить C/AL код в подходящих триггерах.

7.2 Создание Связей между Таблицами

Как было указано в разделе «Введение в Разработку Приложений в Среде C/SIDE», при проектировании реляционных баз данных выделяют три вида связей между таблицами:

- Связь «Один-ко-Многим»
- Связь «Многие-ко-Многим»
- Связь «Один-к-Одному»

Наиболее часто используется связь «Один-ко-Многим». Если при проектировании вам потребовалась связь «Многие-ко-Многим», то это сигнализирует о наличие проблемы в проектировании. Обычно связь «Многие-ко-Многим» преобразуют к двум связям вида «Один-ко-Многим». Связь «Один-к-Одному» часто является нежелательной, ее можно избежать, объединив две таблицы. Для дополнительной информации о проектировании баз данных см. раздел Рекомендуемая Литература по Проектированию Баз Данных.

Для Чего Используются Связи?

Если в базе данных существуют таблицы, содержащие связанные данные, между ними можно установить связь. Связь между таблицами устанавливается на основании одного или нескольких полей, которые содержат одинаковые значения в связанных записях. Часто у таких полей одинаковые наименования в обеих таблицах. Связь между таблицами устанавливается чтобы:

- Выполнять проверку значений.
- Использовать функцию Выбор (Lookup) данных из других таблиц.
- Автоматический перенос изменений из одной таблицы в другие связанные таблицы.

Связи между Таблицами и Свойство TableRelation

Связи между таблицами определяются с помощью свойства *TableRelation*. Данное свойство очень гибкое. Оно позволяет задавать как простые, так довольно сложные связи между таблицами. Типичная простая связь состоит из идентификатора таблицы и иногда идентификатора поля. Сложные связи обычно предваряются условным выражением и содержат фильтры. Ниже приведен синтаксис свойства *TableRelation*:

<TableRelation> ::=

<TableNo>[.<FieldNo>] [WHERE (<TableFilters>)] |

IF (<Conditions>) <TableNo>[.<FieldNo>]

[WHERE(<TableFilters>)] ELSE <TableRelation>

<Conditions> ::=

<TableFilters>

<TableFilters>::=

[<TableFilter> {,<TableFilter>}]

<TableFilter>::=

<DstFieldNo>=CONST(<FieldConst>) |

<DstFieldNo>=FILTER(<Filter>)

Где:

Символ	Примечание
<TableNo>	Указывается связанная таблица
<FieldNo>	Указывается поле в связанной таблице
<Conditions>	Для связи между таблицами может существовать условие
<TableFilters>	Список фильтров таблицы
<TableFilter>	Фильтры таблицы могут быть как константами, так и выражениями-фильтрами.
<DstFieldNo>	Фильтруемое поле (поле на которое накладывается фильтр).
<Filter>	Выражение-фильтр 10|20..30.

DRAFT

Создание Базовых Связей между Таблицами

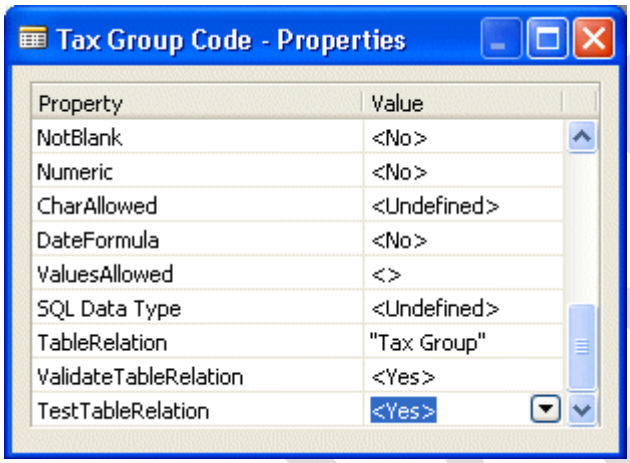
Создавая связь между таблицами, вы можете ввести ее в свойство **TableRelation** вручную или щелкнуть кнопку AssistButton. Обычно вводятся базовые связи, такие как: <TableNo>[.<FieldNo>] непосредственно в окно ***Properties,*** а кнопка AssistButton используется для более сложных связей между таблицами, в которых есть условия и фильтры. Далее будет показано, как создать базовую связь между таблицами, введя ее прямо в окно ***Properties***, а в следующем разделе будет показано, как для то же самое сделать с помощью кнопки AssistButton.

Чтобы создать базовую связь между таблицами следует:

1. Открыть Object Designer (SHIFT+F12), затем щелкнуть кнопку **Table**, чтобы открыть список доступных таблиц.

2. Выбрать таблицу, для которой требуется создать связь с другой таблицей и щелкнуть кнопку **Design**. Откроется окно **Table Designer**.

3. Установить курсор в поле, для которого требуется создать связь. Щелкните Вид, Properties (SHIFT+F4), чтобы открыть окно ***Properties***:



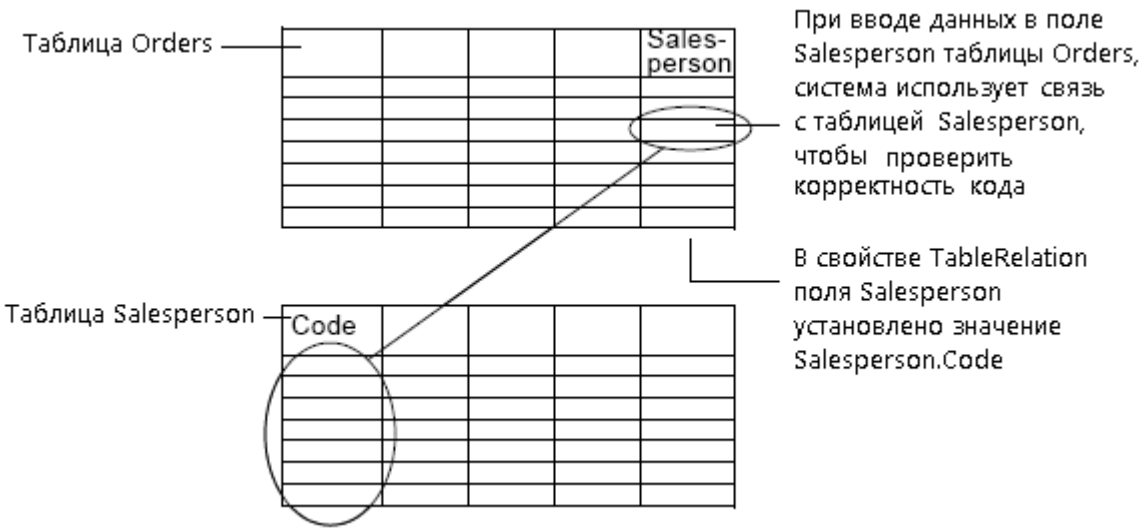
На данном рисунке показаны свойства поля **Tax Group Code** таблицы ***Tax Detail*** (322).

4. Введите связь между таблицами прямо в поле **Value** для свойства *TableRelation*. Простые связи между таблицами используют следующий синтаксис: <TableNo>.[<FieldNo>].

Пример

Предположим, что в таблице ***Orders*** хранятся заказы, а в таблице ***Salesperson*** хранятся имена всех менеджеров компании. В таблицу ***Orders*** можно добавить поле Salesperson, которое позволит определить менеджера. Установив связь между этими двумя таблицами, можно обязать систему проверять, содержит ли поле **Salesperson** таблицы ***Orders*** корректный код.

DRAFT



Пример

Представим, что в базе данных существуют таблица ***Vendors,*** для хранения всех поставщиков и таблица ***Currency Code***. Можно создать связь между полем **Currency Code** в таблице ***Vendors*** и таблицей ***Currency Code***. Это позволит пользователям выбирать (F6) корректные коды валют.

Более того, если в таблице ***Currency Code*** изменить код одной из валют, система автоматически распространит это изменение по таблицам, содержащим ссылку на этот код.

Создание Связей между Таблицами с Помощью AssistButton

В среде C/SIDE доступен специальный инструмент, который упрощает создание сложных связей между таблицами. Сложные связи между таблицами включают предварительное условие, а также используют фильтры.

Чтобы создать связь между таблицами с использованием AssistButton следует:

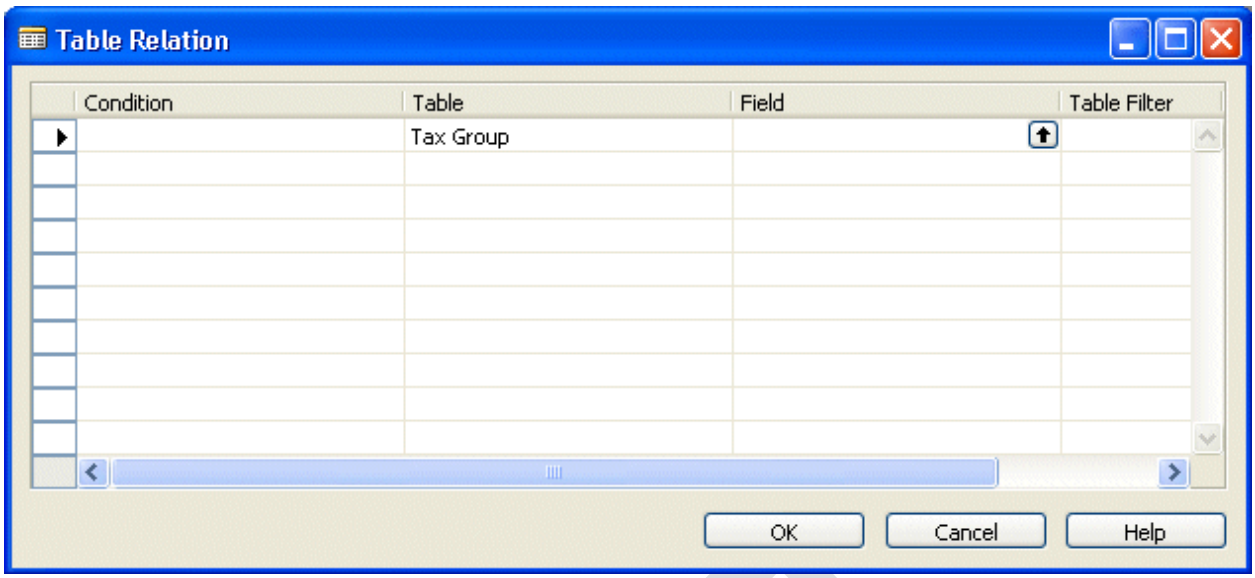
1. Открыть Object Designer (SHIFT+F12), затем щелкнуть кнопку **Table**, чтобы открыть список доступных таблиц.

2. Выбрать таблицу, для которой требуется создать связь с другой таблицей и щелкнуть кнопку **Design**. Откроется окно **Table Designer**.

3. Установить курсор в поле, для которого требуется создать связь. Щелкните Вид, Properties (SHIFT+F4), чтобы открыть окно ***Properties***:

4. Щелкнуть кнопку AssistButton в свойстве *TableRelation*. C/SIDE отобразит окно ***Table Relation***:

DRAFT



На данном рисунке изображено окно ***Table Relation*** для поля **Tax Group Code** таблицы ***Tax Detail*** (322).

5. Заполните поле **Condition** подходящим значением фильтра таблицы. В качестве примера можно посмотреть свойства *TableRelation* в различных таблицах, где тип выбираемого значения зависит от значений других полей (например, поле **No.** таблицы **Sales Header**, которое зависит от поля **Type**).

6. В поле **Table** введите название таблицы, с которой должна быть установлена связь, ее можно выбрать из списка. В поле **Field** указывается наименование поля, его также можно выбрать из списка полей таблицы, указанной в поле **Table**.

При необходимости, можно указать фильтр таблицы (для таблицы указанной в поле **Table**), для этого используется поле **Table Filter**.

Поддержка Связей между Таблицами в SQL Server

Значение свойства *TableRelation* Dynamics NAV может быть представлено в SQL Server как ограничение внешнего ключа. Если этот параметр активирован, связи между таблицами Microsoft Dynamics NAV (ограничения внешних ключей) будут доступны внешним программным средствам.

Чтобы включать и отключать связи между таблицами на SQL Server следует использовать параметр **Поддерживать Связи** на закладке **Интеграция**, доступной из окна **Новая БД** или **Изменить БД**. Для дополнительной информации о данном параметре см. руководство *Installation & System Management: SQL Server Option for Microsoft Dynamics NAV*.

Требования

Существует ряд требований, которые должны быть выполнены, перед тем как свойство *TableRelation* сможет быть представлено в SQL Server.

Чтобы поддерживать связи между таблицами:

- Поля, между которыми устанавливаются связи, должны быть одного типа данных. Тоже относится к полям, указанным в поле **Table Filter**. Типы данных Text и Code совместимы, если у них одинаковая длина.
- Значение свойства *SQL Data Type* для полей с типом данных Code должно быть одинаковым в обеих таблицах.
- При использовании фильтров таблицы, следует применять только тип фильтра *FIELD*, фильтры таблиц типа *CONST* и *FILTER* не могут создаваться в SQL Server.

DRAFT

- Условные связи, в SQL Server будут представлены несколькими связями, по одной для каждого условия.

Синхронизация

Свойство *TableRelation* связи в SQL Server автоматически синхронизируются при создании или изменении таблицы. Однако в некоторых случаях может потребоваться ручная синхронизация связей.

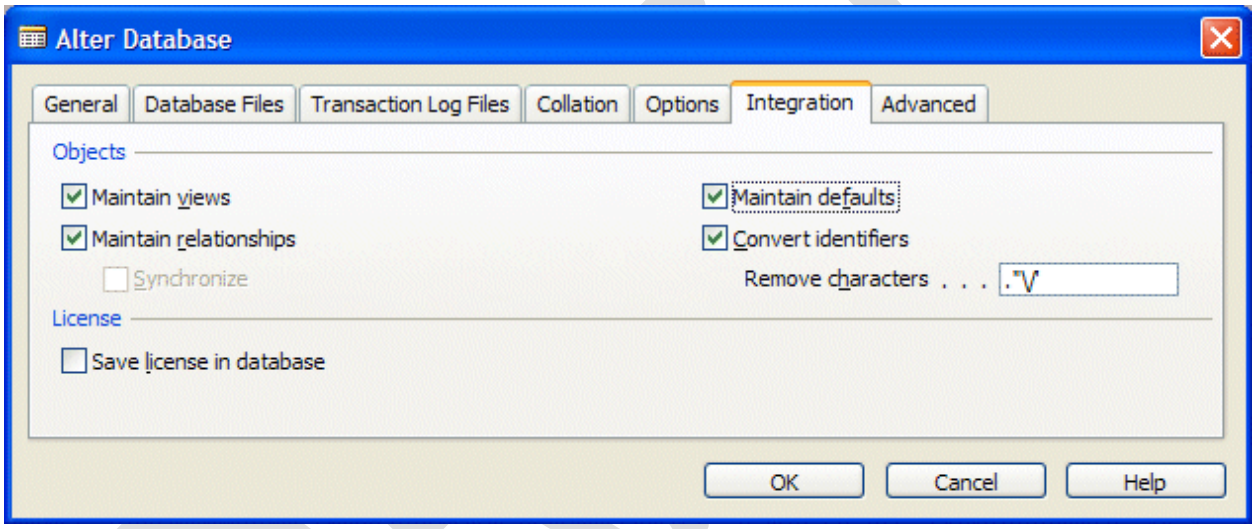
Например:

- Удаление таблицы в Object Designer.
- Восстановление резервной копии.
- Импорт fob-файла.

Чтобы вручную инициировать процесс синхронизации следует:

1. Щелкнуть Файл, База Данных, Изменить. Появится окно **Изменить БД**.

2. Перейти на закладку **Интеграция**.



3. Установить флаг в поле **Synchronize** и щелкнуть кнопку OK.

Данное поле доступно для изменения только когда из-за противоречий в свойстве *TableRelation* требуется синхронизация связей между таблицами.

Если в процессе синхронизации возникла ошибка, система отобразит сообщение об ошибке, в котором будет указано, что для определенной таблицы указаны некорректные связи. Чтобы исправить данную ошибку надо изменить свойство *TableRelation* указанной таблицы и повторно синхронизировать связи.

> **Примечание**
>
> Связи таблиц не генерируются и не поддерживаются при импорте txt-файлов.

7.3 Изменение Таблиц, Содержащих Данные

При проектировании таблицы, задается набор содержащихся в ней полей. Часто может потребоваться изменить структуру таблиц, после того как они некоторое время использовались. Обычно это проявляется в добавлении или удалении полей, а также изменения наименований полей или их типов данных.

> **Примечание**
>
> Среда C/SIDE гарантирует, что при изменении структуры таблиц, содержащих данные, никакие данные не будет потеряны.

Правила, Используемые при Изменении Таблиц

Таблицы, не содержащие данные, можно изменять когда угодно. Но если таблица содержит данные, существует ряд ограничений. Ниже приведена таблица, в которой изложены общие правила:

Изменение	Правило
Изменение наименования поля	Изменять наименование поля можно когда угодно
Изменение номера поля	Изменять номер поля можно когда угодно. При этом Dynamics NAV должен пройти по всем объектам приложения, чтобы обновить ссылки на данное поле. Этот процесс может быть довольно длительным.
Изменение типа данных	Изменять тип данных для поля можно, только в том случае если в нем не хранятся данные. Из этого правила есть одно исключение: изменение типа данных с Code на Text возможно даже если в полях содержаться данные.
Добавление поля в таблицу	Добавлять поля в таблицу можно когда угодно
Удаление поля из таблицы	Чтобы удалить поле из таблицы нужно удалить из него все данные. Также необходимо удалить ссылки на данное поле из других таблиц, форм и отчетов.
Изменение длины текстового поля	Увеличивать длину текстового поля можно когда угодно. Уменьшение длины текстового поля зависит от данных, которые хранятся в данном поле. Минимальная возможная длина текстового поля определяется самым длинным из хранимых в нем значений.

7.4 Связанные Объекты (Linked Objects)

В Dynamics NAV можно создать таблицу, определенную как объект SQL Server (пользовательская таблица, системная таблица или представление данных, которые уже существуют в текущей базе данных).

Определение Свойства Таблицы Linked Object

При создании или изменении Описания Таблицы в Table Designer следует установить в свойстве *LinkedObject* значение Да, чтобы связать таблицу с объектом SQL Server. После изменения значения свойства *LinkedObject* на Да, становится доступным свойство *LinkedInTransaction.*

Значение свойства *LinkedInTransaction* должно быть установлено Нет, если таблица Dynamics NAV ссылается на представление данных, которое основано на внешних объектах или связанных серверах (linked server).

Свойство *LinkedInTransaction* позволяет читать и изменять из связанного сервера источника данных (linked server data sources), таких как Excel, Access или другой SQL Server. Такой доступ не контролируется транзакциями Dynamics NAV. Это означает, что если транзакция Dynamics NAV прерывается, все изменения, выполненные во внешних объектах, сохранятся.

Примечание

Если база данных запущена в режиме доступа Single User, вы не сможете запускать таблицы, для которых в свойстве *LinkedInTransaction* установлено значение Нет.

DRAFT

Создание Описания Таблицы Dynamics NAV

Далее будет приведены описания создания различных видов таблиц Dynamics NAV, в зависимости от значения свойств *LinkedObject* и *LinkedInTransaction*. Чтобы создавать Описания Таблиц вам должна быть назначена роль базы данных - *db_owner*.

Чтобы создать не связанную таблицу:

- Установите значение свойства *LinkedObject = Нет*.
- При сохранении данной таблицы, будет создана таблица SQL Server, владельцем которой является *db_owner*, имя таблицы будет таким, как было указано при ее создании в Dynamics NAV (при необходимости к нему будет добавлено наименование фирмы).
- Если объект с таким наименованием уже существует появиться сообщение об ошибке и таблица не будет сохранена.

Чтобы создать связанный объект, который контролируется транзакциями:

- Установите значение свойства *LinkedObject = Да*.
- Установите значение свойства *LinkedInTransaction = Да*.
- Таблица будет сохранена без проверки. Dynamics NAV проверит существования соответствующего объекта SQL, а также его совместимость с Описанием Таблицы Dynamics NAV только при подключению к таблице.

Чтобы создать связанный объект, который не контролируется транзакциями:

- Установите значение свойства *LinkedObject = Да*.
- Установите значение свойства *LinkedInTransaction = Нет*.
- Таблица будет сохранена без проверки. Dynamics NAV проверит существования соответствующего объекта SQL, а также его совместимость с Описанием Таблицы Dynamics NAV только при подключению к таблице.

Удаление Описания Таблицы Dynamics NAV

Если в свойстве *LinkedObject* установлено значение *Нет*

- Объект SQL Server удаляется, если он является пользовательской таблицей.
- Объект SQL Server не удаляется, если он является системной таблицей или представлением данных. Объект может быть системной таблицей или представлением данных, только при использовании внешних инструментов. Для связи с системными таблицами или представлениями данных, следует установить в свойстве *LinkedObject* значение Да.

Если в свойстве *LinkedObject* установлено значение Да:

- Объект SQL Server не удаляется, это значит, что таблица Dynamics NAV была создана при значении свойства *LinkedObject = Нет*, а затем значение свойства было изменено на Да.

При подключении к данным в связанном объекте:

- Если свойство *LinkedInTransaction* = Да, все подключения к связанному объекту выполняются под контролем транзакций Dynamics NAV.
- Если свойство *LinkedInTransaction* = Нет, все подключения к связанному объекту выполняются независимо от транзакций Dynamics NAV.

Требования к Связанным Объектам

При использовании связанных объектов следует принимать во внимание следующее:

DRAFT

- Наименование объекта SQL Server должно включать префикс в виде наименования фирмы, разделитель ($), и должно полностью совпадать с названием таблицы Dynamics NAV.
- Для создания обычных таблиц, нужно иметь роль *db_owner* в текущей базе данных.
- Как и в случае с обычными таблицами Dynamics NAV объекты должны существовать в текущей базе данных и их владельцем должен быть пользователь с ролью *db_owner*. Представления данных SQL Server могут использоваться для доступа к объектам за пределами текущей базы данных (в том числе на других серверах) или владельцами которых являются другие пользователи.
- Dynamics NAV автоматически предоставит права доступа SQL на объекты, таким образом доступ к ним выполняется также как к обычным таблицам.
- Определение таблиц SQL подключенного объекта должно совпадать с определением таблицы Dynamics NAV.
- Представления данных нельзя обновлять (update) в SQL Server. В SQL Server 2000, можно создать Instead-Of триггеры, в которых указать логику обновления представления данных. Данная логика не является частью логики Dynamics NAV.

Правила Определения Совместимости

Существует ряд правил, которые необходимо помнить при использовании связанных объектов:

- Все поля объекта должны иметь совместимые типы данных с полями таблицы Dynamics NAV, соответствующие поля определяются на основании наименований полей. Информацию по совместимости типов данных смотри Совместимость Типов Данных.
- Ни для каких типов объектов не могут использоваться поля SumIndexFields.
- Если объект является пользовательской таблицей, у него должно быть задано ограничение первичного ключа, в который входят такие же поля как в первичный ключ в Dynamics NAV. Эти поля должны иметь одинаковые наименования.
- Если объект является представлением данных или системной таблицей, для него должен быть определен первичный ключ. При необходимости, также должны быть созданы вторичные ключи. Эти ключи используются только в Dynamics NAV. Они не оказывают никакого влияния на представления данных или системные таблиц. Важно, чтобы данные в поле, определенном как первичный ключ, были уникальными. Физических ограничений для представлений видов или системных таблиц SQL Server не существует. Однако Dynamics NAV сортирует данные согласно заданному первичному ключу. Dynamics NAV полагается на данную уникальность для корректной идентификации и сортировки записей.
- Если объект является представлением данных, он может иметь только одно поле с типом timestamp, таблица Dynamics NAV не обязана содержать соответствующее BLOB поле. Поле с типом timestamp должно существовать для пользовательских таблиц.
- Поле IDENTITY может использоваться в пользовательских таблицах или представлениях данных, Dynamics NAV проигнорирует данное поле, при вставке записей в таблицу. Аналогично, вычисляемые поля в пользовательских таблицах также игнорируются. Для представлений данных, если планируется использовать операции вставки, нельзя использовать поля, основанные на вычисляемых полях таблиц.
- Нельзя создавать связи с временными таблицами SQL Server.
- Многоязыковые представления данных не создаются и не поддерживаются для связанных объектов. Подробнее смотри раздел “Creating and Maintaining Databases” в руководстве *Installation & System Management: SQL Server Option for Microsoft Dynamics NAV*.

После того как для объекта была создана связь, Dynamics NAV обрабатывает его как обычную таблицу. Но в зависимости от типа объекта SQL Server может запрещать выполнение некоторых операций. Например, не обновляемые представления данных не могут быть обновлены из

DRAFT

Dynamics NAV, в этом случае SQL Server отобразит сообщение об ошибке. Также ограничена возможность по изменению структуры объекта из Dynamics NAV, подробнее данные ограничения описаны в следующем разделе.

Изменение Таблиц Dynamics NAV с Определенным Связанным Объектом

Таблицы Dynamics NAV с определенным связанным объектом могут быть изменены в соответствии со следующими правилами:

- Нельзя изменять наименование таблицы или наименование фирмы.
- Нельзя переименовывать поля таблицы.
- Можно добавлять новые поля при условии, что они существуют в представлении данных, также можно удалять существующие поля. В обоих случаях представление данных SQL Server не изменяется.
- Первичный и вторичные ключи могут быть изменены. Также могут быть добавлены новые ключи, а существующие удалены.
- Для полей Dynamics NAV можно изменить типы данных при условии, сохранения совместимости с данными в полях представления данных.
- Для связанной пользовательской таблицы можно выполнять любые изменения, которые применимы к обычным таблицам, созданным в Dynamics NAV.
- Если для таблицы Dynamics NAV изменено свойство *DataPerCompany,* это приведет к попытке установить связь с новым объектом. Этот новый объект будет основываться на новом наименовании фирмы. Предыдущий связанный объект SQL Server не будет связан с таблицей.
- Свойство *LinkedObject* можно изменяться с Да на Нет только для пользовательских таблиц.

Подключение к Объектам в Других База Данных или Связанных Серверах

Для объектов расположенных вне текущей базы данных или сервера можно организовать доступ и Dynamics NAV создав соответствующее представление данных (view) в текущей базе данных. Также можно создать представление данных на основании связанных серверов, которые могут подключаться к разнородным источникам данных. Например, это могут быть таблицы Oracle, Access или лист данных Excel.

Чтобы выполнить подключение к объектам в других базах данных или на связанных серверах, нужно следовать следующим правилам:

- Установить в свойстве *LinkedInTransaction* значение Нет, чтобы иметь возможность использовать представления данных, ссылающиеся на объекты за пределами текущей базы данных. Возможность изменять данные в этих объектах зависит от поставщика данных (data providers).
- Чтобы иметь возможность подключаться к другим базам данных или связанным серверам, нужно иметь роль *db_owner* в текущей базе данных.
- Все права доступа к объектам в другой базе данных или связанном сервере должны быть определены вне Dynamics NAV и назначены соответствующим логинам SQL Server.
- Если связанный объект ссылается на представление данных, которое получает данные из другой базы данных, хранящейся на том же сервере, такое представление данных должно обрабатываться, словно оно получает данные со связанного сервера.

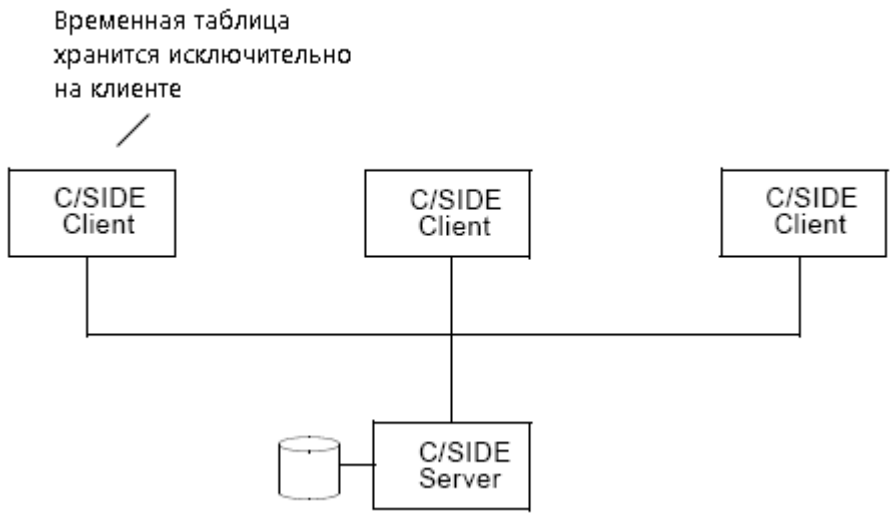
DRAFT

ГЛАВА 8 СПЕЦИАЛЬНЫЕ ТАБЛИЦЫ C/SIDE

Кроме обычных таблиц базы данных, в среде C/SIDE есть еще три типа таблиц, которые используются для специальных целей в приложениях C/SIDE. Это временные (temporary), системные (system) и виртуальные (virtual) таблицы. Временные таблицы используются как хранилище временной информации во время работы, два других типа таблиц генерируются системой и предоставляют различную информацию о ее текущем состоянии.

Данная глава посвящена специальным таблицам среды C/SIDE и объяснению, как их можно использовать в своих проектах.

- Что такое временная таблица?
- Что такое системная таблица?
- Что такое виртуальная таблица?



8.1 Что Такое Временная Таблица?

Понятие временной таблицы относится к временной переменной, которая содержит таблицу. Временная таблица используется как буфер для хранения данных в ваших C/AL программах. Если вы не знакомы с программированием на языке C/AL, ознакомьтесь с ГЛАВА 16 ВВЕДЕНИЕ В ЯЗЫК C/AL.

Вы можете делать с временными таблицами все то, что и с обычными таблицами базы данных. Отличие временных таблиц от обычных заключается в следующем:

- Временные таблицы не хранятся в базе данных. Временная таблица содержится в памяти до тех пор, пока она не будет закрыта.
- Принципы транзакций, применяемые к обычным таблицам базы данных, не используются для временных таблиц. Если вы не знакомы с принципами транзакций, см. раздел Транзакции Записи и Восстановления (Write Transactions and Recovery).

Преимуществом использования временных таблиц является то, что работа с временной таблицей выполняется на клиенте. Это уменьшает нагрузку на сеть и на сервер.

При необходимости выполнения большого количества операций над данными в определенной таблице базы данных, можно загрузить эти данные во временную таблицу для последующего изменения. Загрузка данных во временную таблицу ускорит процесс, так как все операции будут выполняться на клиенте.

Определение и Использование Временных Таблиц

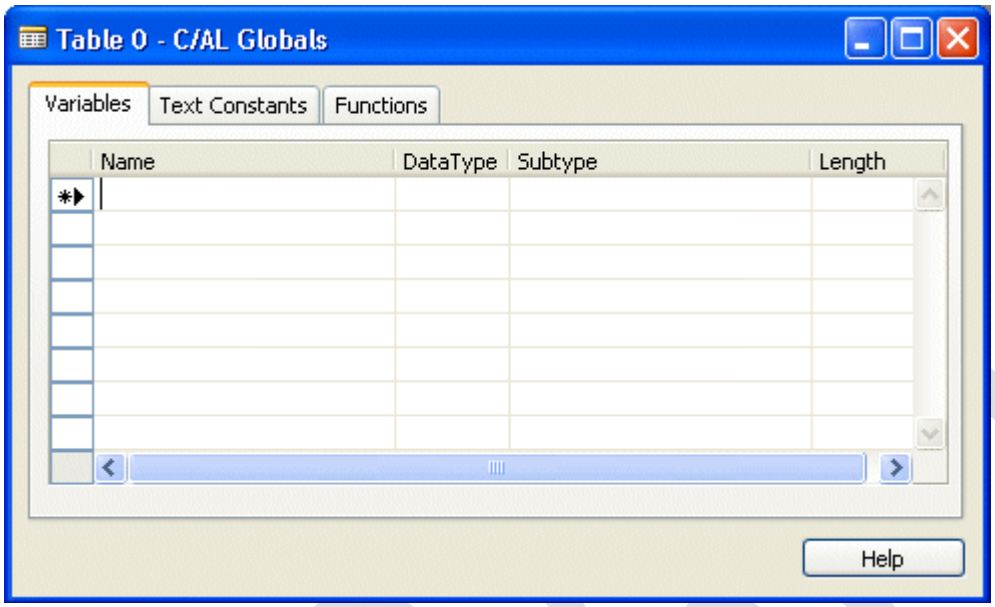
Перед тем как использовать временную таблицу в C/AL коде ее нужно определить. Переменная, содержащая временную таблицу, определяется точно также как любая другая глобальная или локальная переменная.

Чтобы определить временную таблицу:

1. Открыть Object Designer (SHIFT+F12) и создать новую таблицу.

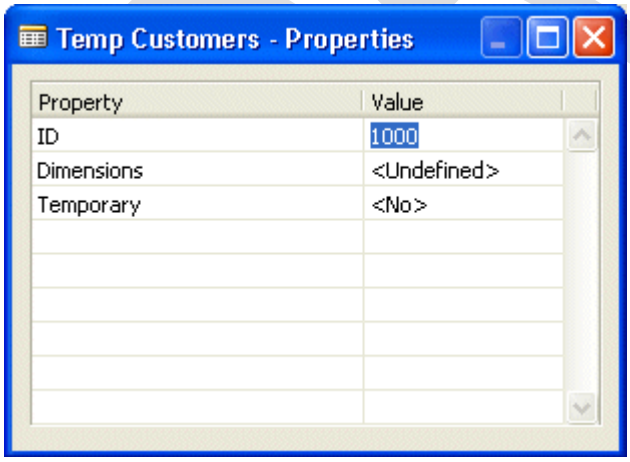
2. Щелкнуть Вид, C/AL Globals или C/AL Locals, в зависимости от того должна быть переменная локальной или глобальной.

Если выбрано C/AL Globals, появится окно ***C/AL Globals***:



3. Ввести наименование переменной для временной таблицы и ввести или выбрать тип данных Record. В поле Subtype выбрать таблицу.

4. Пока курсор установлен на строке, в которой определена переменная для временной таблицы, щелкнуть Вид, Properties (SHIFT+F4). Откроется окно ***Properties***:



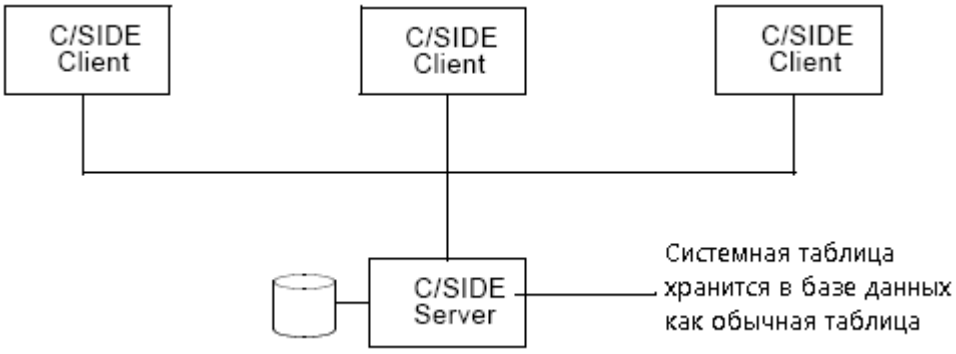
5. Изменить значение свойства *Temporary = Да*.

Временная таблица создана, теперь ее можно использовать в C/AL коде. К ней можно применять фильтры и выполнять поиск как в обычной таблице базы данных.

8.2 Что Такое Системная Таблица?

Системная таблица хранится в базе данных, как и обычная таблица. Однако в отличие от обычных таблиц, системные таблицы создаются автоматически. Информация в данных таблицах тесно

связана с СУБД, которая использует данные таблицы для управления, например, системой безопасности и правами доступа. Информацию в данных таблицах можно читать, добавлять, изменять и удалять.



В среде C/SIDE восемь системных таблиц:

- ***User***
- ***Member Of***
- ***User Role***
- ***Permission***
- ***Windows Access Control***
- ***Windows Login***
- ***Company***
- ***Database Key Groups***

Первые шесть связаны с системой безопасности.

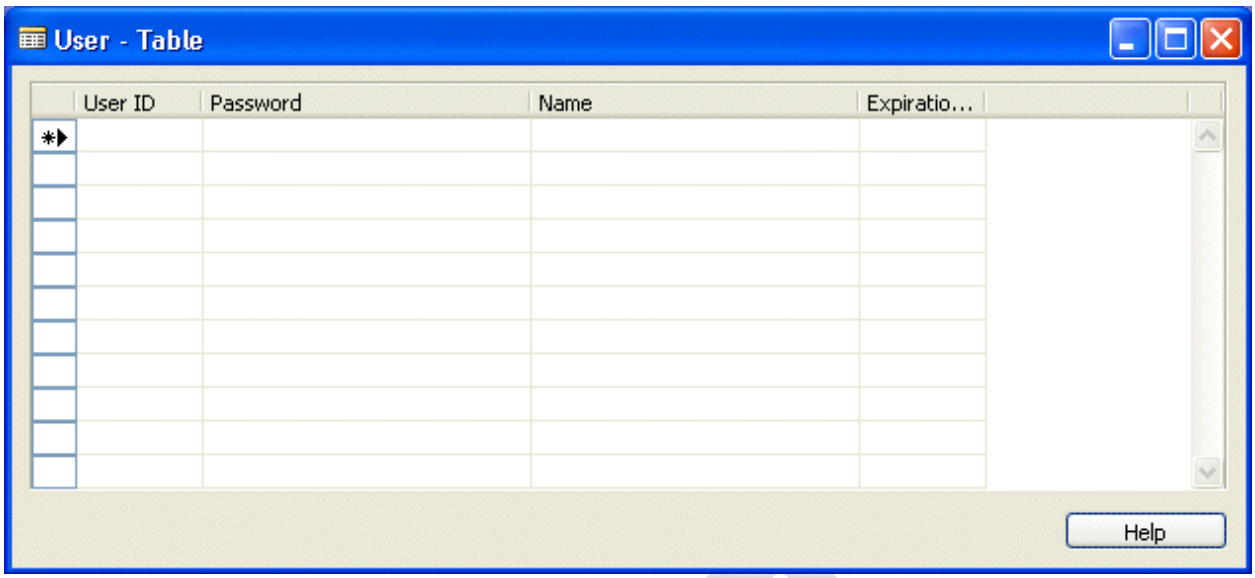
> **О правах доступа**
>
> Чтобы вставлять изменять или удалять информацию из таблиц ***User***, ***Member Of***, ***User Role***, ***Permission***, ***Windows Access Control*** и ***Windows Login*** нужно иметь права доступа не ниже, прав доступа пользователей, чьи права доступа подвергаются изменениям. Это значит, что вы можете назначать только такие права доступа какими обладаете сами.

Далее будет предоставлен обзор этих системных таблиц. Подробную информацию о системе безопасности Dynamics NAV, см. в руководстве *Installation & System Management* для используемого варианта сервера.

Системная Таблица User (Пользователь)

Системная таблица ***User*** предоставляет обзор всех кодов пользователей, созданных для пользователей в базе данных. Каждая запись в таблице ***User*** соответствует одному коду пользователя.

DRAFT



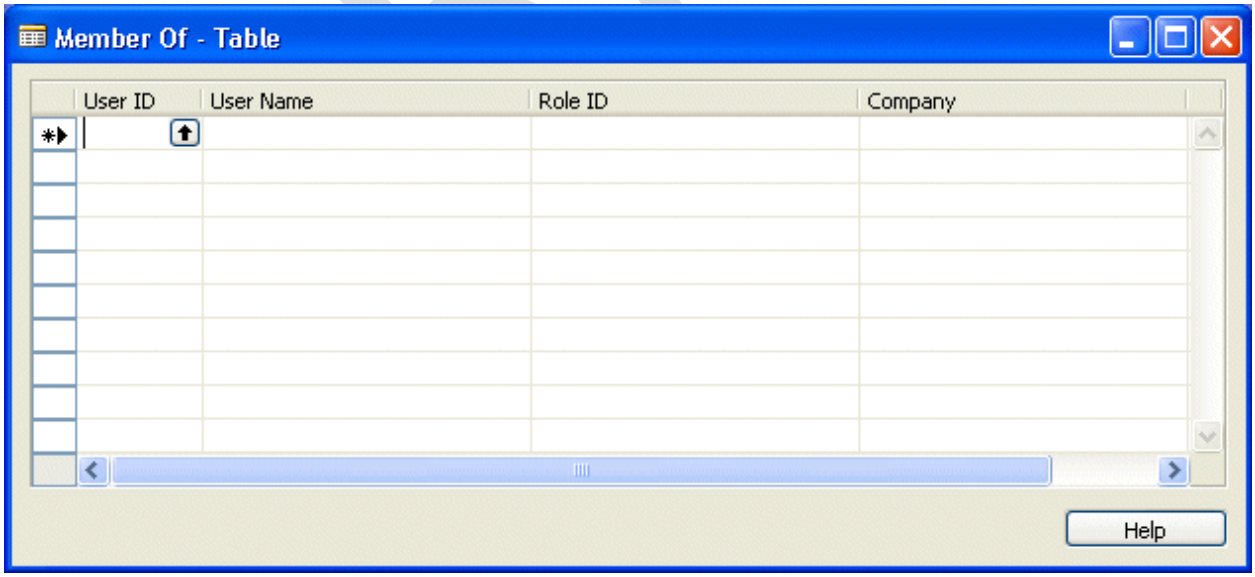
В таблице ***User*** содержится информация о пароле (на форме отображается в зашифрованном виде), настоящее имя пользователя и до какого времени можно пользоваться указанным регистрационным именем. Новый код пользователя можно создать, введя в таблицу соответствующие данные. Так же из данной таблицы можно удалить код пользователя, удалив запись (для этого надо обладать подходящими правами доступа)

Удаление Записей

При удалении записей из системной таблицы ***User*** автоматически удаляются связанные записи в системной таблице ***Member Of***.

Системная таблица Member Of (Член)

Системная таблица ***Member Of*** предоставляет информацию о том, какие группы (роли) назначены пользователю. Для каждого кода пользователя можно назначить неограниченное количество ролей.



Системная Таблица User Role (Роль Пользователя)

Системная таблица ***User Role*** представляет информацию о ролях пользователей в текущей базе данных. Роль пользователя определяется набором прав доступа. Права доступа для каждой роли пользователя задаются в системной таблице ***Permissions***.

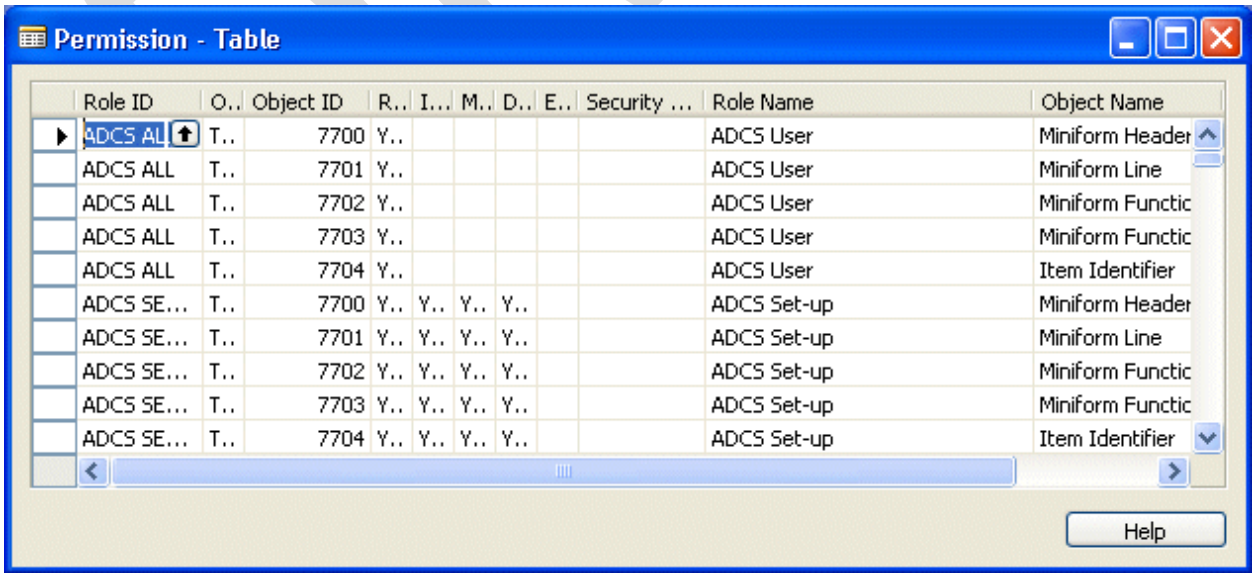
Удаление Записи

При удалении записи в системной таблице ***User Role*** автоматически удаляются связанные записи в системных таблицах ***Member Of*** и ***Permission***.

Системная Таблица Permission (Права Доступа)

Системная таблица ***Permission*** используется для указания разрешенных действий для роли пользователя. Права доступа определяются для объектов, таким образом можно задать права доступна для каждой таблицы, формы и т.д. Можно указать, что данная роль пользователя не имеет прав доступа (пусто), имеет полные (Да) или Косвенные права на выполнение следующих действий:

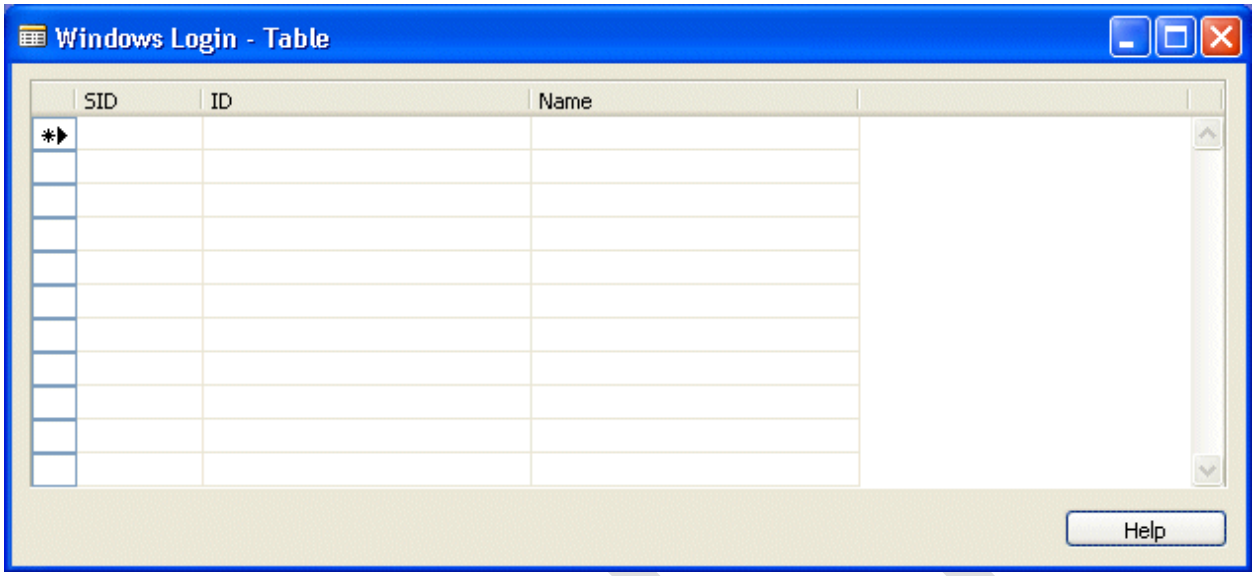
- Чтение
- Создание/Вставка
- Изменение
- Удаление
- Запуск



Системная Таблица Windows Login (Регистрационное Имя Windows)

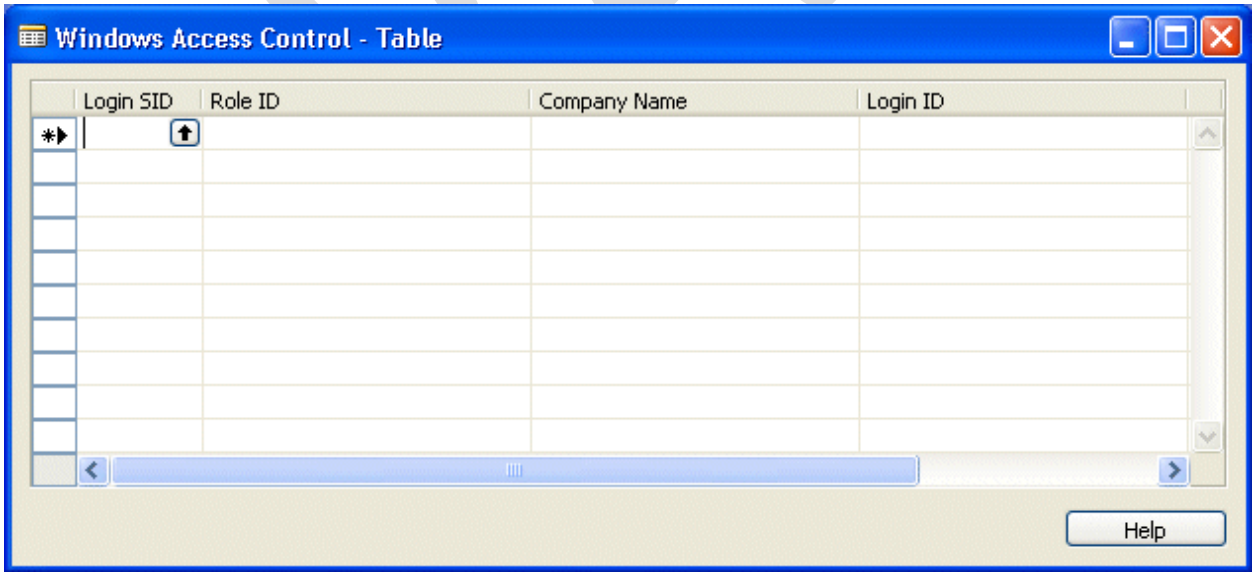
Системная таблица ***Windows Login*** предназначена для указания пользователей Windows и их групп, которые могут входить в систему. Только указанные в таблице пользователи Windows или

пользователи, входящие в указанные группы могут входить в систему. У каждого пользователя или группы Windows есть уникальный идентификатор безопасности (SID). Поле ID заполняется на основании наименования пользователя или группы, определенной на основании SID. Поле **Name** в настоящий момент не используется.



Системная Таблица Windows Access Control (Контроль Доступа Windows)

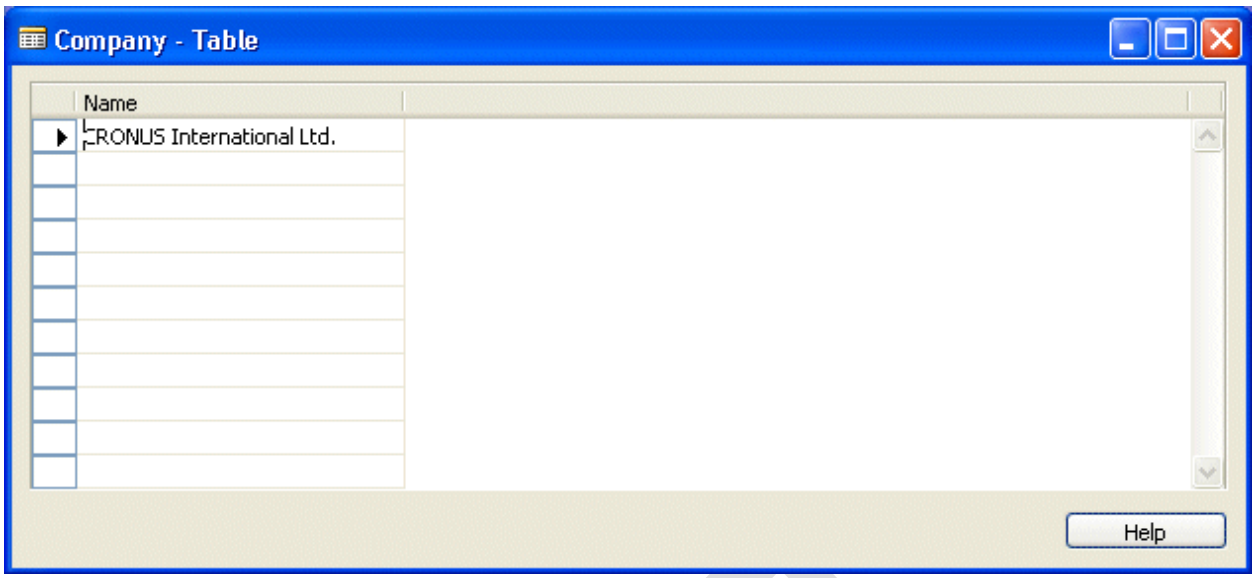
Системная таблица ***Windows Access Control*** используется для управления правами доступа пользователей и групп пользователей Windows к Dynamics NAV. Каждый пользователи или группа Windows имеют уникальные идентификаторы безопасности (SID). Дополнительно, для каждого пользователя или группы назначены роли пользователя, которые связаны с правами доступа к фирме в Dynamics NAV. Информация отображается в полях **Login ID** и **Role Name** и основывается на значении SID и роли пользователя соответственно.



Системная Таблица Company (Фирма)

В системной таблице ***Company*** представлены все фирмы текущей базы данных. Каждая фирма представлена отдельной записью. Создать новую фирму можно вставив новую запись в данную таблицу. Фирму можно удалить из базы данных, удалив запись в таблице ***Company***.

При удалении фирмы, удаляются все таблицы фирмы и все права доступа к данной фирме.



Системная Таблица Database Key Groups (Группы Ключей БД)

Системная таблица ***Database Key Groups*** предоставляет обзор групп ключей, определенных в текущей базе данных. Каждая запись относится к одной группе ключей.

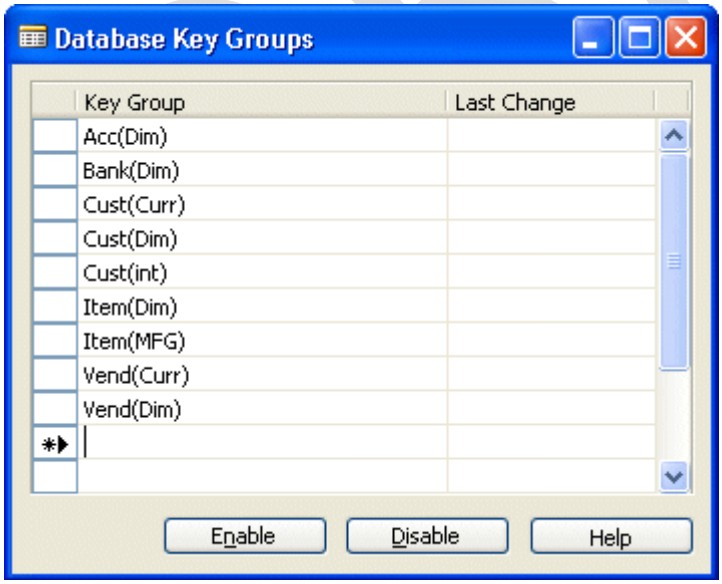
> **Примечание о Группах Ключей**
>
> Указывая, что ключ входит в группу ключей, вы получаете возможность Включать и Выключать различные комбинации ключей в таблицах, включая или выключая группу ключей.

Чтобы включить или выключить группу ключей:

1. Щелкните Файл, База Данных, Информация. Появится окно **Информация о БД**.

2. Щелкните кнопку **Таблицы**, чтобы открыть окно **Информация БД (Таблицы)**.

3. Щелкните кнопку **Группы Ключей**, чтобы открыть окно **Группы Ключей БД**:



4. Выберите группу ключей и щелкните кнопку **Выключить** или **Выключить**, чтобы включить или выключить группу ключей.

8.3 Что Такое Виртуальная Таблица?

Виртуальная таблица содержит информацию предоставляемую системой. Среда C/SIDE позволяет подключаться к ряду виртуальных таблиц. Они работают также как обычные таблицы базы данных, но в них нельзя изменять информацию. Эта информация только для чтения. Второе отличие заключается в том, что виртуальные таблицы не хранятся в базе данных, а создаются системой во время работы.

Когда Используются Виртуальные Таблицы

Виртуальные таблицы предоставляют единообразный интерфейс к различной информации. Так как работа с виртуальными таблицами аналогична работе с обычными таблицами, для доступа к ним используются такие же методы, как и для доступа к обычным таблицам. Например, вы можете использовать фильтры, чтобы получить подмножество или диапазон дат или целых чисел.

Виртуальные таблицы предоставляют доступ к следующей информации:

- Целые числа в диапазоне от -1.000.000.000 до 1.000.000.000.
- Даты за указанный период.
- Обзор файлов операционной системы.
- Обзор логических дисков.
- Отслеживание запросов от клиента к базе данных.
- Обзор пользователей подключенных к базе данных в текущий момент.
- Обзор файлов операционной системы, в которых хранится база данных.

Обзор Виртуальных Таблиц

C/SIDE включает следующие виртуальные таблицы:

- Date,
- Integer,
- File,
- Drive,
- Monitor,
- Session,
- Database File,
- Table Information,
- Field,
- Key,
- Server,
- Windows Object,
- Windows Group Member,
- SID - Account ID,
- User SID

DRAFT

Использование Виртуальных Таблиц

Первые из указанных таблиц предоставляют легкий доступ к датам, целым числам, информации о файлах в операционной системе и логических дисках компьютера.

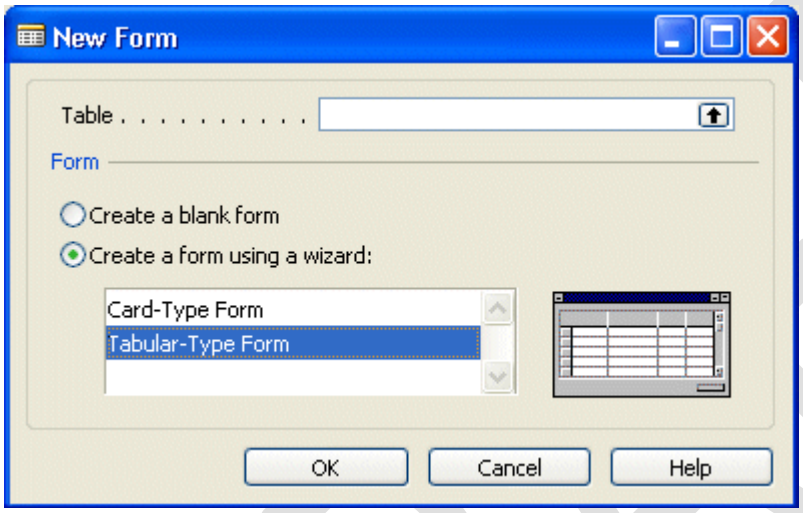
Так как виртуальные таблицы не хранятся в базе данных, они не видны из Object Designer. Чтобы посмотреть значения в виртуальной таблице нужно создать табличную форму, которая будет основана на данной таблице.

Чтобы посмотреть значения в виртуальной таблице следует:

1. Открыть Object Designer (SHIFT+F12).

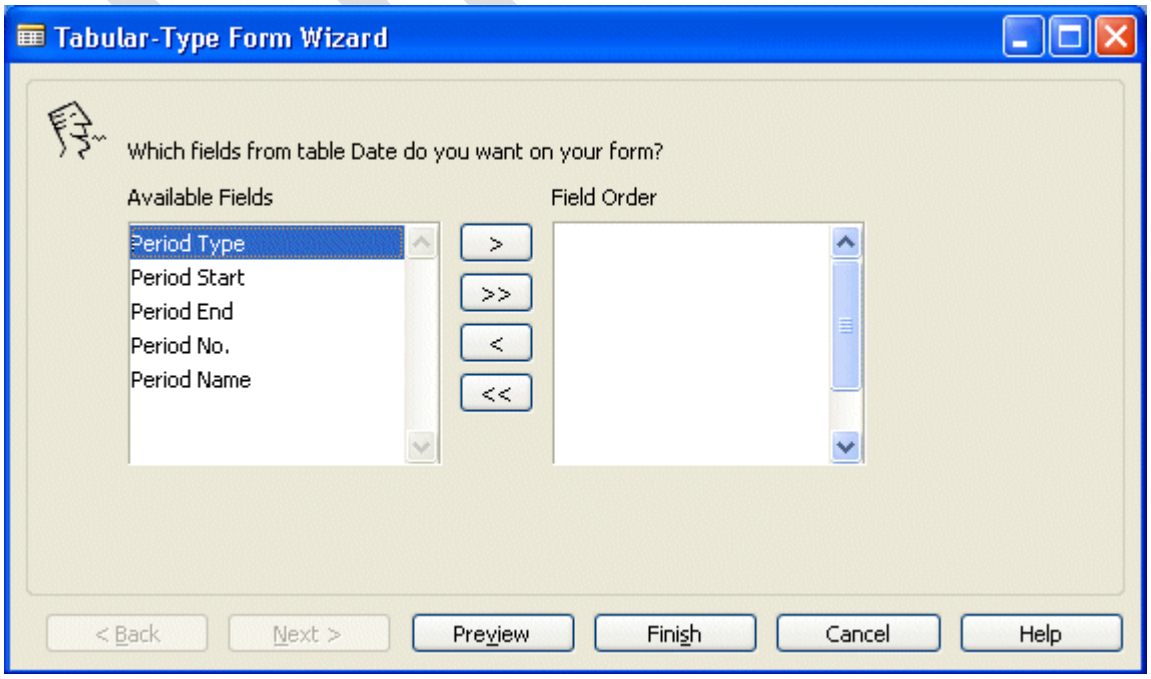
2. Щелкнуть кнопку **Form**, затем кнопку **New**, чтобы открыть окно ***New Form***.

3. Выберите вариант **Create a form using a wizard** и убедитесь что выбран тип формы Tabular-Type Form:



4. В поле **Table** введите наименование виртуальной таблицы, на основании которой будет основана форма. Как вариант можно использовать кнопку Выбор (F6), чтобы открыть список всех таблиц в базе данных. Виртуальные таблицы находятся в наивысшем диапазоне номеров (2000000001 – 2000000203). В данном примере будет использована таблица ***Date***.

5. Щелкните кнопку OK. Появится окно ***Tabular-Type Form Wizard***:



6. Щелкните кнопку >>, чтобы переместить все доступные поля в поле **Field Order**.

7. Щелкните кнопку Preview, чтобы посмотреть таблицу и ее содержимое. Как вариант можно щелкнуть кнопку Finish и сохранить форму.

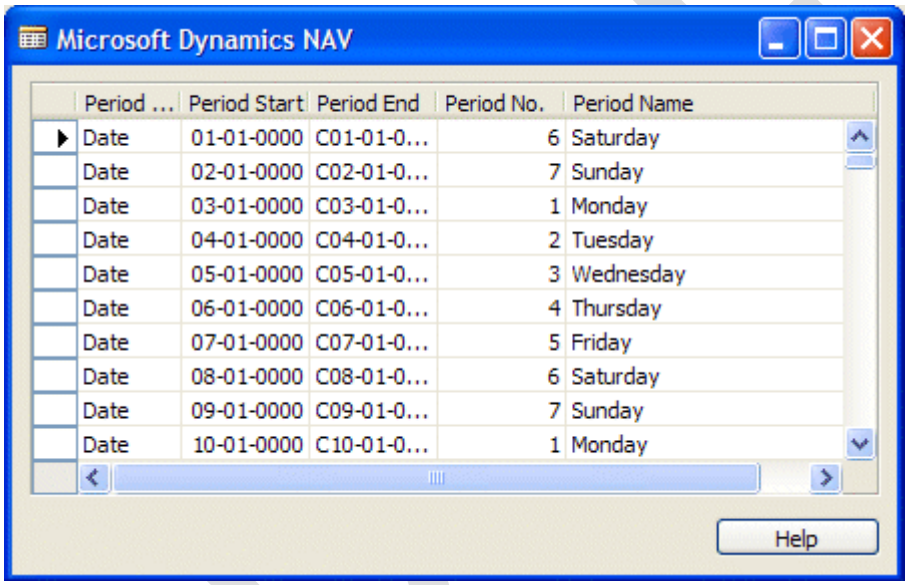
Этот метод используется для просмотра остальных виртуальных таблиц.

Виртуальная Таблица Date (Дата)

Данная виртуальная таблица предоставляет простой доступ к Дням, Неделя, Месяцам, Кварталам и Годам. Поля виртуальной таблицы ***Date***:

Поле	Примечание
Period Type	День, Неделя, Месяц, Квартал, Год
Period Start	Дата первого дня периода
Period End	Дата последнего дня периода
Period No.	Номер периода
Period Name	Название периода

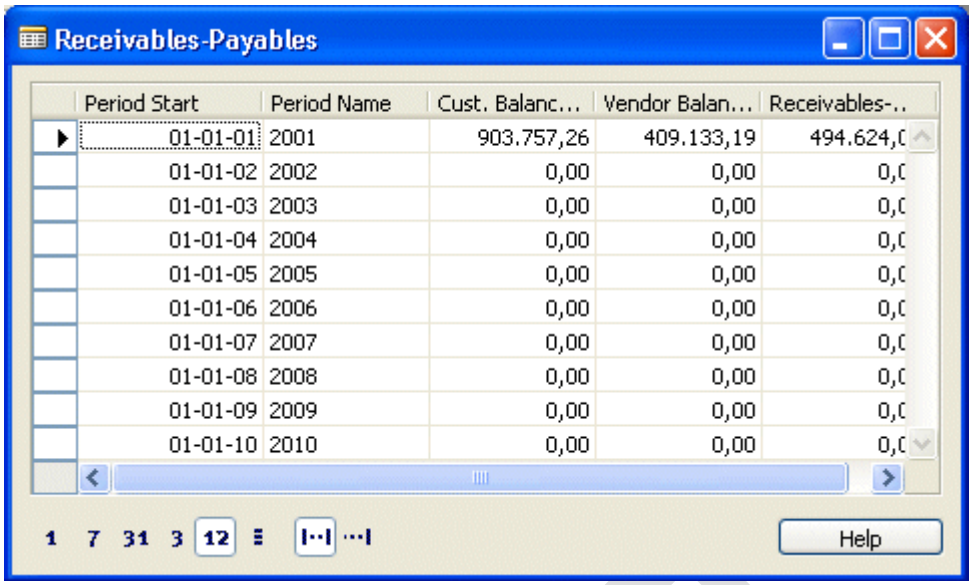
На следующем рисунке показано как выглядит виртуальная таблица ***Date***. Для каждого типа периодов (День, Неделя и т.д.) существует множество записей в таблице ***Date***:



На поля **Period Type**, **Period Start**, и **Period End** можно накладывать фильтры, чтобы легко получать доступ к подмножествам дней, недель, месяцев, и т.д. для использования в формах или отчетах.

Пример

Виртуальная таблица ***Date*** часто используется для предоставления диапазонов дат, типичный пример – форма Поступления-Платежи.



Виртуальная Таблица Integer (Целое)

Данная виртуальная таблица включает целые числа в диапазоне от -1,000,000,000 на 1,000,000,000. Виртуальная таблица ***Integer*** содержит только одно поле:

Поле	Примечание
Integer	Целые числа в диапазоне с -1.000.000.000 до 1.000.000.000

Накладывая фильтры на данную виртуальную таблицу, можно легко получить подмножество чисел, которое используются для организации циклов в отчетах.

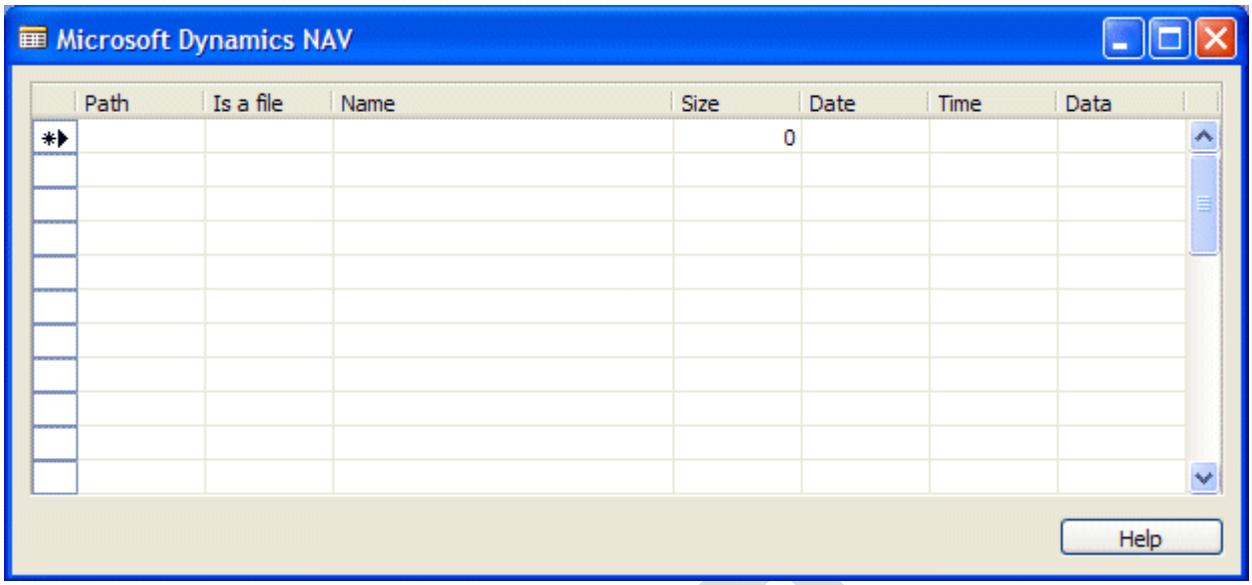
Виртуальная Таблица File (File)

Данная виртуальная таблица предоставляет обзор файлов в каталоге жесткого диска. Виртуальная таблица ***File*** содержит следующие поля:

Поле	Примечание
Path	Устанавливая фильтры на данное поле, можно указать файлы какого должны показываться.
Is a File	Значение Да показывает, что данная запись относится к файлу, значение Нет показывает, что запись является каталогом.
Name	Название файла или каталога
Size	Размер файла в байтах
Date	Дата последнего изменения файла
Time	Время последнего изменения файла
Data	Поле BLOB, содержащее файл

Чтобы получить доступ к виртуальной таблице **File** необходимо создать табличную форму:

DRAFT

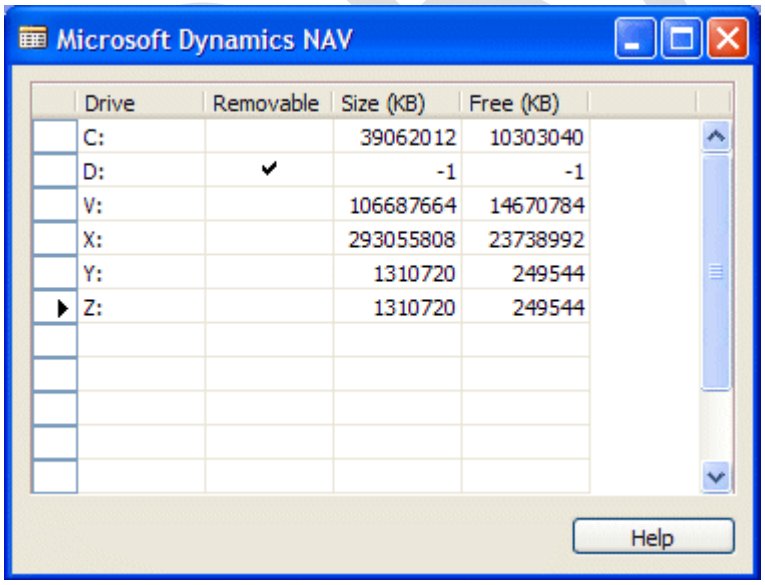


Виртуальная Таблица Drive (Drive)

Эта виртуальная таблица предоставляет обзор логических дисков компьютера. Виртуальная таблица ***Drive*** содержит следующие поля:

Поле	Примечание
Drive	Наименование диска, например, A: или D:
Removable	Показывает, является ли устройство съемным (карта памяти или CD-ROM)
Size (KB)	Общий объем диска
Free (KB)	Свободное место на диске

Чтобы получить доступ к виртуальной таблице **Drive** необходимо создать табличную форму:



Примечание

Если в устройстве со съемными носителями нет диска, поля **Size (KB)** и **Free (KB)** содержат значение *-1*.

Другие виртуальные таблицы обычно используются администратором системы. Они предоставляют администратору систему информацию о подключенных в данный момент пользователях, а также о текущем состоянии системы.

Виртуальная Таблица Monitor (Монитор)

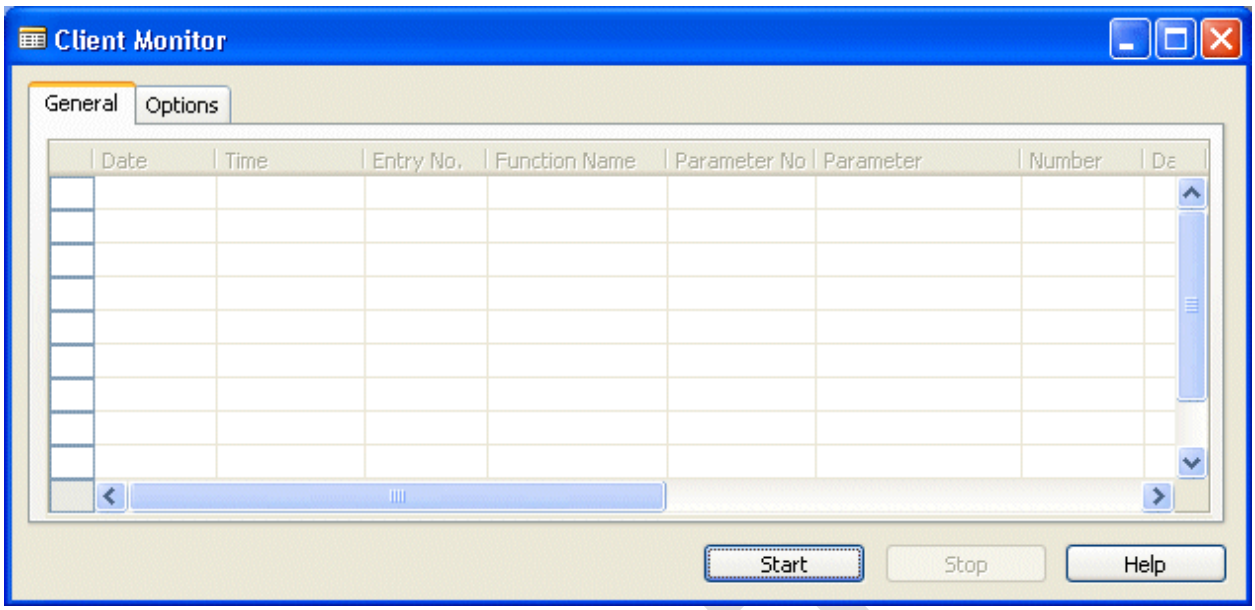
Данная виртуальная таблица отслеживает все запросы, выполненные клиентом Dynamics NAV, к таблицам в базе данных. Обычно виртуальную таблицу ***Monitor*** используют C/AL программисты, чтобы узнать длительность выполнения определенной операции. Они могут использовать информацию из данной виртуальной таблицы для оптимизации кода.

Виртуальная таблица ***Monitor*** содержит следующие поля:

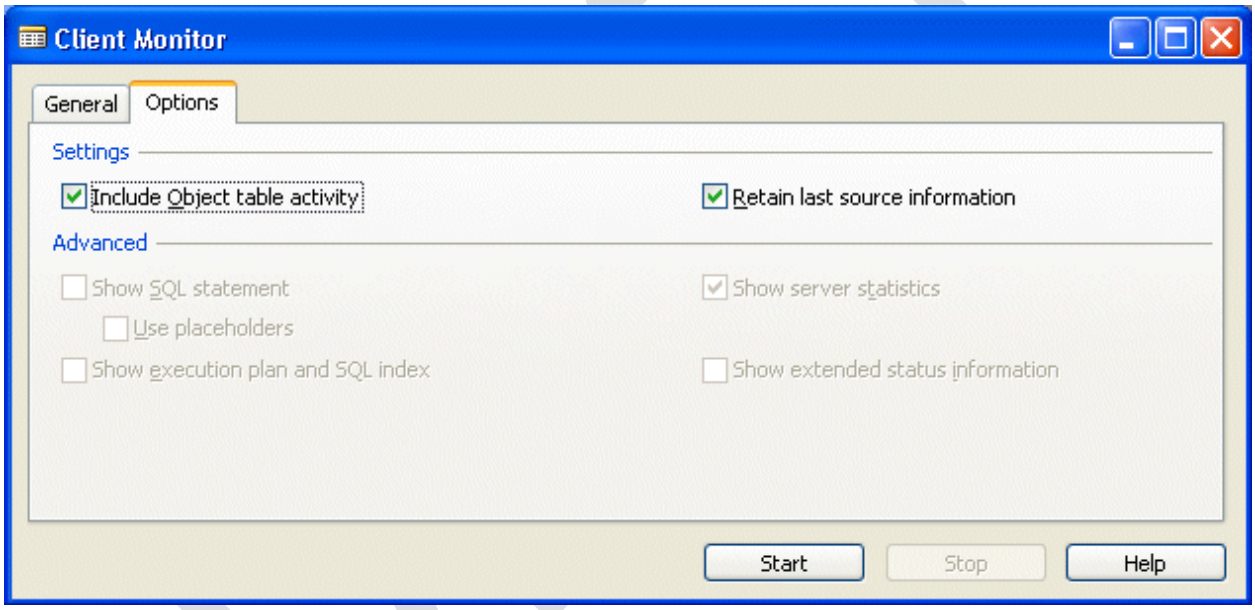
Поле	Примечание	Возможные значения
Entry No.	Последовательные номера, которые увеличиваются для каждого запроса	от 1 до 2^{31}-1
Function Name	Тип запроса	LOCKTABLE, DELETE, MODIFY, INSERT, ELETEALL, Create Key, Delete Key, Redesign Table, FIND/NEXT, CALCSUMS, CALCSUMS (Slow), COMMIT, Delete Table, Create Database, Close Database, Open Database, Delete Database, Expand Database, Get Table Statistics, COUNT, Get Database Statistics, Optimize Key, Login, Read Database Block, Read BLOB, Insert BLOB, Delete BLOB, Clear Old Versions, Get Database Free Percent, Preload Database Block, и т.д.
Parameter No.	Число параметров	Зависит от числа параметров
Parameter	Наименование параметров	Table, Key, Order, Filter, Search Method, Search Result, Records Found, Sum, CPU (ms), Records Read, Sum Intervals, Records Deleted, Records Modified, Disk Reads, Disk Writes, Record, Wait, SumIndexFields, BLOB Field, Commit, User ID, File Name, Source Object, Source Trigger/Function, Source Line No., Source Text, Record Count, Timeout Status, Time Out (ms)
Number	Если параметр является числом, его значение отображается в данном столбце	Любые числовые значения
Data	Все нечисловые значения отображаются в данном столбце	Любые строковые значения

Чтобы получить доступ к виртуальной таблице ***Monitor*** щелкните Сервис, Монитор Клиента:

DRAFT



В окне Монитор Клиента содержится две закладки. На закладке **Параметры** можно указать вид информации собираемой Монитором Клиента.



Закладка **Параметры** содержит следующие параметры (параметры из группы Дополнительно доступны при использовании SQL Server):

Параметр	Поведение при выборе
Включать Таблицу Активности Объектов	В Монитор Клиента записываются операции, выполняемые над таблицей ***Object***.
Сохранять информацию о последнем источнике	Даже для функций не связанных с выполнением C/AL кода в Монитор Клиента записываются параметры Source Object, Source Trigger/Function, Source Line No. и Source Line
Показ выражения SQL	Отображается выражение SQL

Параметр	Поведение при выборе
Использовать placeholders	Вместо значений фильтров в выражениях SQL используются '?'
Показ статистики сервера	Собирается статистика по 'Server Time', 'Logical Reads' и 'Records Read'.
Показ плана выполнения и индекса SQL	Планы выполнения SQL (для большинства выражений) отображается как перевернутое дерево: Plan Step(Object)[n,p];... где, для узла (node) n родитель (parent) p. Пример: Computer Scalar[2,1];Clustered Index Seek(User$0)[4,2] Дерево будет следующим: 1 – Root ---------2 -- Computer Scalar ----------4 -- Clustered Index Seek(User$0) Индекс SQL отображается как список полей.
Показ расширенной информации о статусе	Для статусов SQL отображается дополнительная информация: internal unique statement ID, reuse status, prepared status, cursor type, optimizer hints, transaction type.

Чтобы использовать Монитор Клиента для отслеживания обращений к серверу:

1. Щелкните Сервис, Монитор Клиента.

2. Откройте закладку **Параметры** и установите требуемые параметры.

3. Щелкните кнопку **Старт**, чтобы активизировать **Монитор Клиента**.

4. На время выполнения анализируемых операций окно **Монитор Клиента** можно закрыть.

5. После выполнения анализируемых операций щелкните Сервис, Монитор Клиента, чтобы снова открыть окно **Монитор Клиента**.

6. Чтобы остановить **Монитор Клиента** щелкните кнопку **Стоп**.

Виртуальная Таблица Session (Сессия)

Данная виртуальная таблица предоставляет информацию о пользователях, подключенных в данный момент к C/SIDE Database Server или SQL Server.

Виртуальная таблица ***Session*** содержит следующие поля:

Поле	Примечание	SQL Server	NDS

DRAFT

Поле	Примечание	SQL Server	NDS
Connection ID	Код подключения	X	X
User ID	Код подключенного пользователя	X	X
My Session	Указывает, принадлежит ли сессия данном пользователю	X	X
Login Time	Время, когда пользователь осуществил вход в систему и начал данную сессию	X	X
Login Date	Дата входа пользователя в систему	X	X
Database Name	Имя базы данных, которая бала открыта в эту сессию	X	X
Application Name	Название приложения подключенного к серверу	X	X
Login Type	Указывается, является ли данная сессия входом Windows или базы данных	X	X
Cache Reads	Количество операций чтения из кэша за данную сессию.(A)		X
Disk Reads	Количество операция чтения с диска за данную сессию.(A)		X
Disk Writes	Количество операций записи на диск за данную сессию.(A)		X
Records Found	Количество найденных записей за данную сессию		X
Records Scanned	Количество сканированных записей за данную сессию		X
Records Inserted	Количество вставленных записей за данную сессию		X
Records Deleted	Количество удаленных записей за данную сессию		X
Records Modified	Количество измененных записей за данную сессию.		X
Sum Intervals	Количество переходов выполненных системой при расчете сумм за данную сессию. Высокое значение указывает на неэффективный ключ		X
Host Name	Наименование рабочей станции, использующей данную сессию	X	
CPU Time (ms)	Кумулятивная величина времени CPU, приходящаяся на эту сессию	X	
Memory Usage (KB)	Число килобайт в кэше процедур, которое в настоящее время отведено для данной сессии	X	
Physical I/O	Кумулятивное число считываний с диска и записи на него, приходящееся на эту сессию	X	
Idle Time	Время, прошедшее с того момента, когда сервер последний раз получил запрос от данной сессии. Если это поле пустое значит, сессия в данный момент выполняет запрос к базе данных. Это может быть длительный запрос, например, перестройка таблиц.	X	X
Blocked	Указывает, заблокирована или нет данная сессия другой сессией.	X	

DRAFT

Поле	Примечание	SQL Server	NDS
Wait Time (ms)	Время, в течение которого сессия находится в состоянии ожидания	X	
Blocking Connection ID	Код подключения заблокировавшего данную сессию	X	
Blocking User ID	Код пользователя, относящийся к подключению, заблокировавшему данную сессию	X	
Blocking Host Name	Имя рабочей станции, использующей подключение, которое заблокировало данную сессию	X	
Blocking Object	Имя объекта SQL, который заблокировал данную сессию.	X	

(A) Если свойство Commitcache = Yes

Важно

Если разработанное решение использует какое-либо поле, недоступное на SQL Server для Dynamics NAV, его следует изменить для возможности запуска на SQL Server. Данные поля не создаются для SQL Server, поэтому если программа попытается получить к ним доступ, появится сообщение об ошибке.

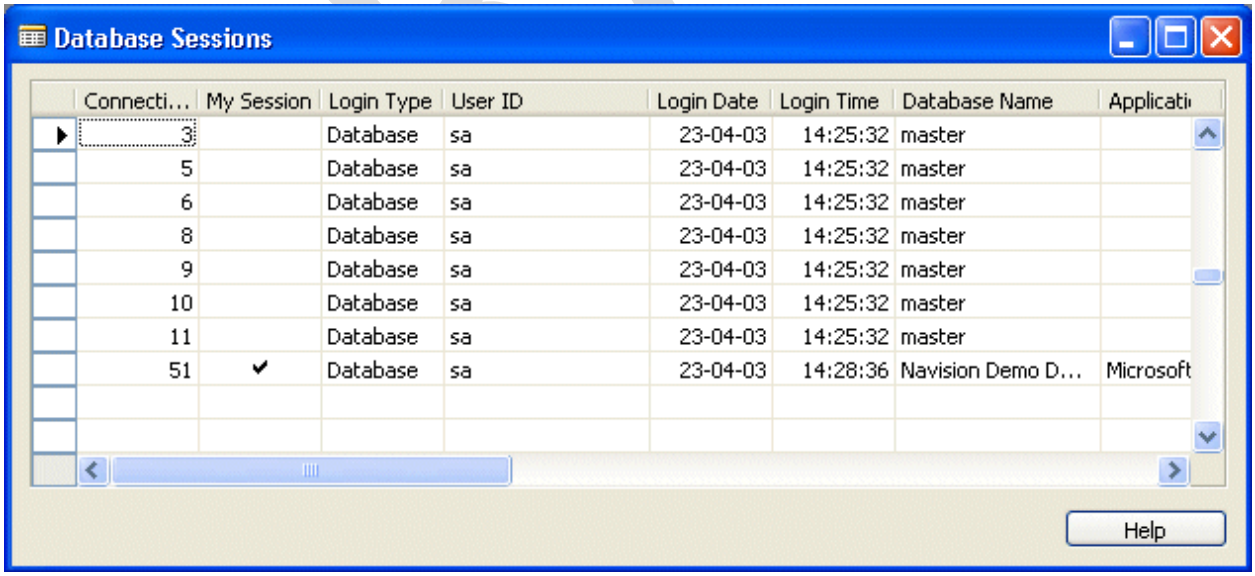
C/SIDE использует виртуальную таблицу ***Session*** для отображения информации о базе данных.

Чтобы получить доступ к виртуальной таблице ***Session***:

1. Щелкните Файл, База Данных, Информация.

2. В окне **Информация о БД** перейдите на закладку **Сессии**.

3. Щелкните кнопку Детализировать (F6) в поле **Текущие Сессии**. Появится окно **Сессии БД**:



Примечание

В зависимости от используемого варианта сервера, окно **Сессии БД** отображает разные поля виртуальной таблицы ***Session***.

Администратор может завершить одну или несколько сессий, выбрав строку и удалив ее. Пользователь будет отключен от сервера и должен будет перезапустить программу, если он хочет продолжать работу. Чтобы завершать сессии, у администратора должны быть права на удаление

данных из виртуальной таблицы ***Session.*** Для SQL Server администратор должен иметь роль SQL Server - *sysadmin* или *processadmin*.

Виртуальная Таблица Database File (Файл БД)

В данной виртуальной таблице представлены файлы операционной системы, которые содержат базу данных. Виртуальная таблица ***Database File*** содержит следующие поля:

Поле	Примечание
No.	Номер файла
File Name	Наименование файла.
Size (KB)	Размер файла в КБ.
Total Reads	Количество операций чтения с момента открытия базы данных.(B)
Mean Read Time (ms)	Среднее время операции чтения (в миллисекундах).(B)
Reads In Queue	Число операций чтения ожидающих в очереди.(A)(B)
Total Writes	Количество операций записи с момента открытия базы данных.(B)
Mean Write Time (ms)	Среднее время операции записи (в миллисекундах).(B)
Writes In Queue	Число операций записи ожидающих в очереди.(A)(B)
Disk Load (%)	Загрузка диска в процентах.(B)

(A) Если свойство Commitcache = Yes

(B) Не доступно в SQL Server для Dynamics NAV.

> **Важно**
>
> Если разработанное решение использует какое-либо поле, недоступное на SQL Server для Dynamics NAV, его следует изменить для возможности запуска на SQL Server. Данные поля не создаются для SQL Server, поэтому если программа попытается получить к ним доступ, появится сообщение об ошибке.

C/SIDE использует виртуальную таблицу ***Database File*** для отображения информации о базе данных.

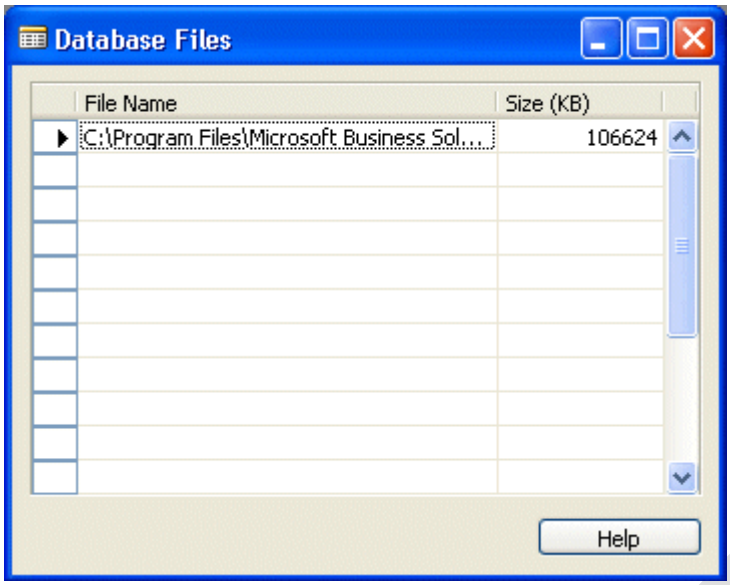
Чтобы получить доступ к виртуальной таблице ***Database File***:

1. Щелкните Файл, База Данных, Информация.

2. Перейдите на закладку **База Данных**.

3. Щелкните кнопку Детализировать (F6) в поле **Имя БД.** Откроется окно **Файлы БД**:

DRAFT



Примечание

В зависимости от используемого варианта сервера, окно **Файлы БД** отображает разные поля виртуальной таблицы ***Database File***.

Виртуальная Таблица Table Information (Таблица Информация)

В данной виртуальной таблице содержится информация о таблицах базы данных. Виртуальная таблица ***Table Information*** содержит следующие поля:

Поле	Примечание
Company Name	Название компании, которой принадлежит таблица.
Table No.	Идентификатор таблицы
Table Name	Наименование таблицы
No. of Records	Количество записей в таблице
Record Size	Средний размер записи в таблице. Рассчитывается как: 1024 * Size(KB) / No. of Records.
Size (KB)	Сколько места занимает таблица в базе данных (в КБ).
Optimization	Данный коэффициент измеряет эффективность хранения данных в таблице Microsoft Dynamics NAV. Эффективность может быть повышена посредством уплотнения данных в таблице. Нажмите кнопку Оптимизировать в нижней части окна.(A)

(A) Не доступно в SQL Server для Dynamics NAV.

Важно

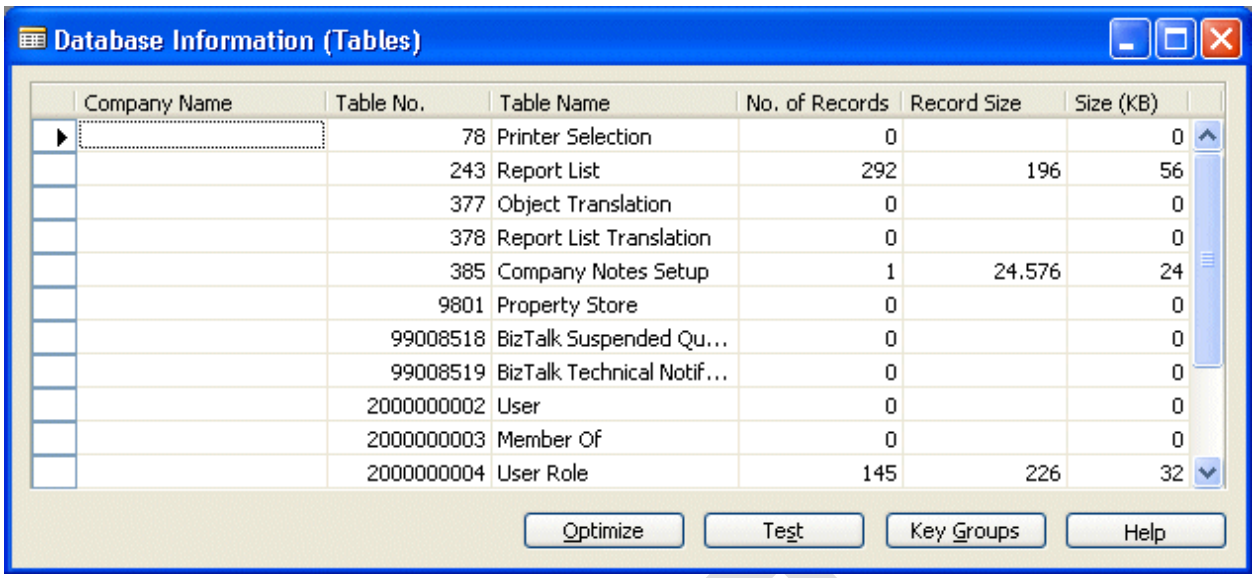
Если разработанное решение использует какое-либо поле, недоступное на SQL Server для Dynamics NAV, его следует изменить для возможности запуска на SQL Server. Данные поля не создаются для SQL Server, поэтому если программа попытается получить к ним доступ, появится сообщение об ошибке.

Чтобы получить доступ к виртуальной таблице ***Table Information***:

1. Щелкните Файл, База Данных, Информация.

2. Щелкните кнопку **Таблицы**. Откроется окно **Информация БД (Таблицы)**:

DRAFT



Примечание

В зависимости от используемого варианта сервера, окно **Информация БД (Таблицы)** отображает разные поля виртуальной таблицы ***Table Information***.

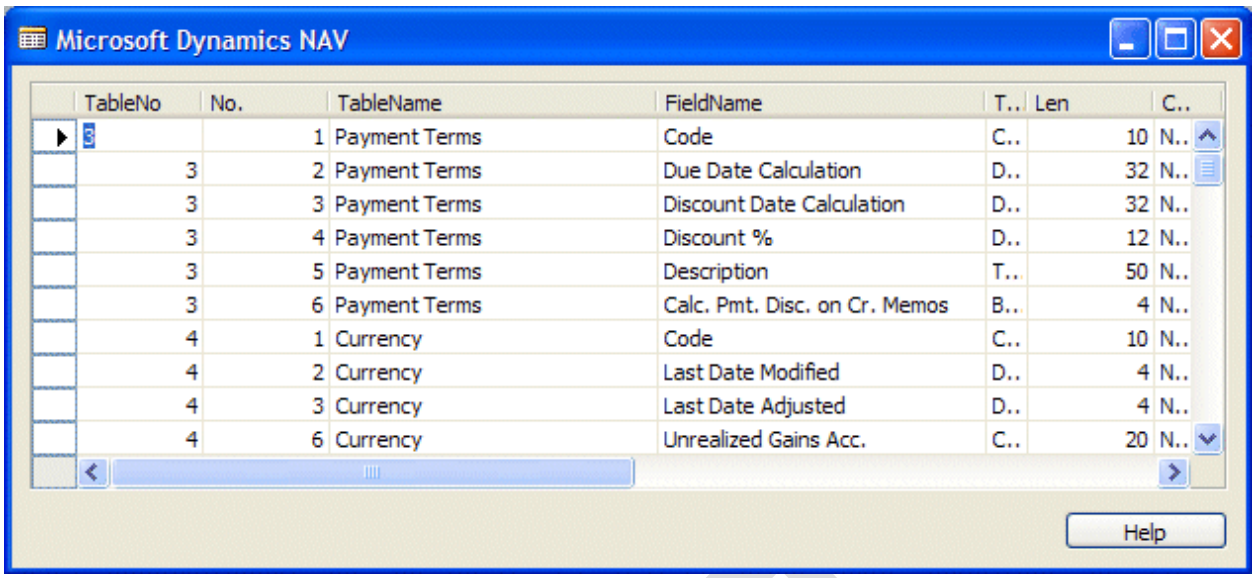
Виртуальная Таблица Field (Field)

В данной виртуальной таблице содержится информация о полях таблиц в базе данных. Виртуальная таблица ***Field*** содержит следующие поля:

Поле	Описание
TableNo	Идентификатор Таблицы
No.	Номер поля
Table Name	Наименование таблицы
FieldName	Наименование поля
Type	Тип данных, назначенный для поля, например, decimal.
Len	Длина поля в байтах
Class	Класс поля, например, FlowField (вычисляемое поле).
Enabled	Является ли поле активным
Type Name	Наименование типа данных назначенного для поля. Для полей Code и Text включается длина в байтах.
Field Caption	Заголовок поля на языке, который установлен в программе.
RelationTableNo	Идентификатор таблицы, на которую ссылается поле.
RelationFieldNo	Номер поля в другой таблице, на которое ссылается данное поле.
SQLDataType	Тип данных SQL, присвоенный для поля типа Code

Чтобы получить доступ к виртуальной таблице ***Field*** необходимо создать табличную форму:

DRAFT



Microsoft Dynamics NAV

TableNo	No.	TableName	FieldName	T..	Len	C..
3	1	Payment Terms	Code	C..	10	N..
3	2	Payment Terms	Due Date Calculation	D..	32	N..
3	3	Payment Terms	Discount Date Calculation	D..	32	N..
3	4	Payment Terms	Discount %	D..	12	N..
3	5	Payment Terms	Description	T..	50	N..
3	6	Payment Terms	Calc. Pmt. Disc. on Cr. Memos	B..	4	N..
4	1	Currency	Code	C..	10	N..
4	2	Currency	Last Date Modified	D..	4	N..
4	3	Currency	Last Date Adjusted	D..	4	N..
4	6	Currency	Unrealized Gains Acc.	C..	20	N..

Help

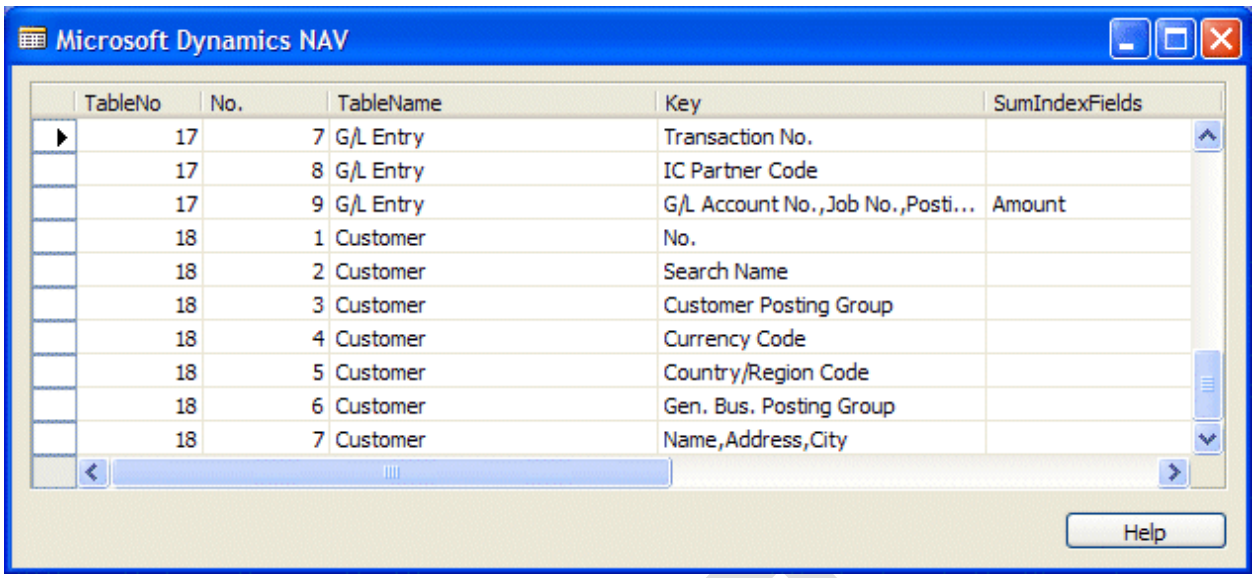
Виртуальная Таблица Key (Key)

В данной виртуальной таблице содержится информация о ключах таблиц в базе данных. Виртуальная таблица ***Key*** содержит следующие поля:

Поле	Описание
TableNo.	Идентификатор таблицы
No.	Номер ключа
TableName	Наименование таблицы
Key	Поля, входящие в ключ
SumIndexField	Поля SumIndexFields, определенные для данного ключа
SQLIndex	Поля, определенные в Dynamics NAV и используемые в качестве индексов SQL Server вместо полей входящих в ключ.
Enabled	Является ли ключ активным. Это поле можно изменять.
MaintainSQLIndex	Активно ли свойство MaintainSQLIndex. Это поле можно изменять.
MaintainSIFTIndex	Активно ли свойство MaintainSIFTIndex. Это поле можно изменять.
Clustered	Активно ли свойство Clustered. Это поле можно изменять.

Чтобы получить доступ к виртуальной таблице ***Key*** необходимо создать табличную форму:

DRAFT

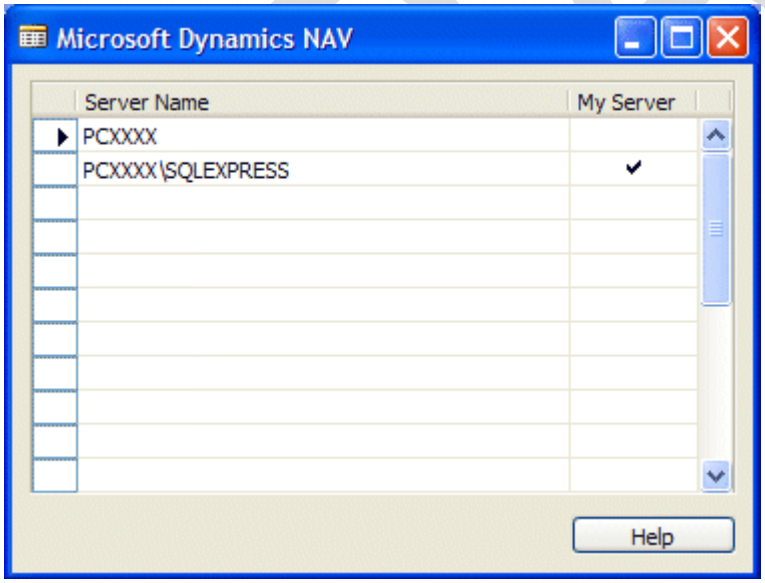


Виртуальная Таблица Server (Server)

Если используется SQL Server для Dynamics NAV, можно получить доступ к виртуальной таблице ***Server***. Виртуальная таблица ***Server*** содержит следующие поля:

Поле	Описание
Server Name	Наименование компьютера, на котором установлен сервер
My Server	Подключены ли вы к данному серверу

Чтобы получить доступ к виртуальной таблице ***Server*** необходимо создать табличную форму:



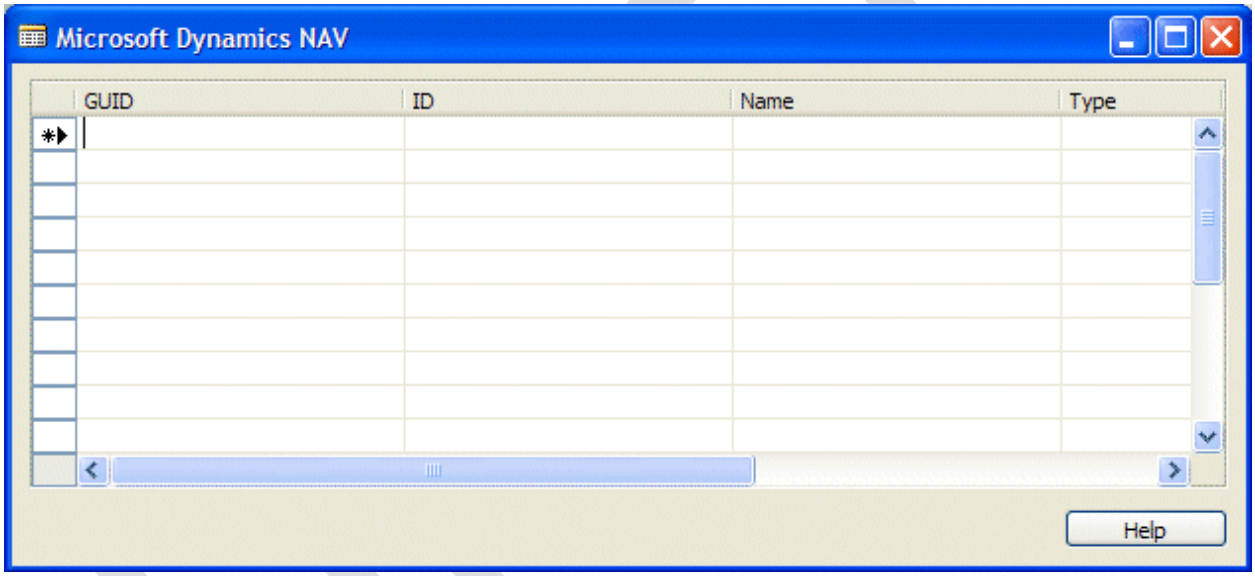
Виртуальная Таблица Windows Object (Объект Windows)

В данной виртуальной таблице представлены пользователи Windows и группы Windows, которые могут быть включены в систему безопасности Dynamics NAV. Данную информацию можно видеть, если клиент включен в Active Directory. Dynamics NAV получает данные из Active Directory. Виртуальная таблица ***Windows Object*** содержит следующие поля:

Поле	Описание

Поле	Описание
GUID	Глобальный уникальный идентификатор (GUID) пользователя или группы Windows.
ID	Идентификатор пользователя Windows или группы Windows.
Name	Наименование объекта Windows. Объект может пользователем Windows или группой Windows.
Type	Тип объекта
SID	Уникальный идентификатор безопасности (SID) пользователя или группы Windows
Distinguished Name	Определяет домен, включающий объект Windows.

Чтобы получить доступ к виртуальной таблице ***Windows Object*** необходимо создать табличную форму:

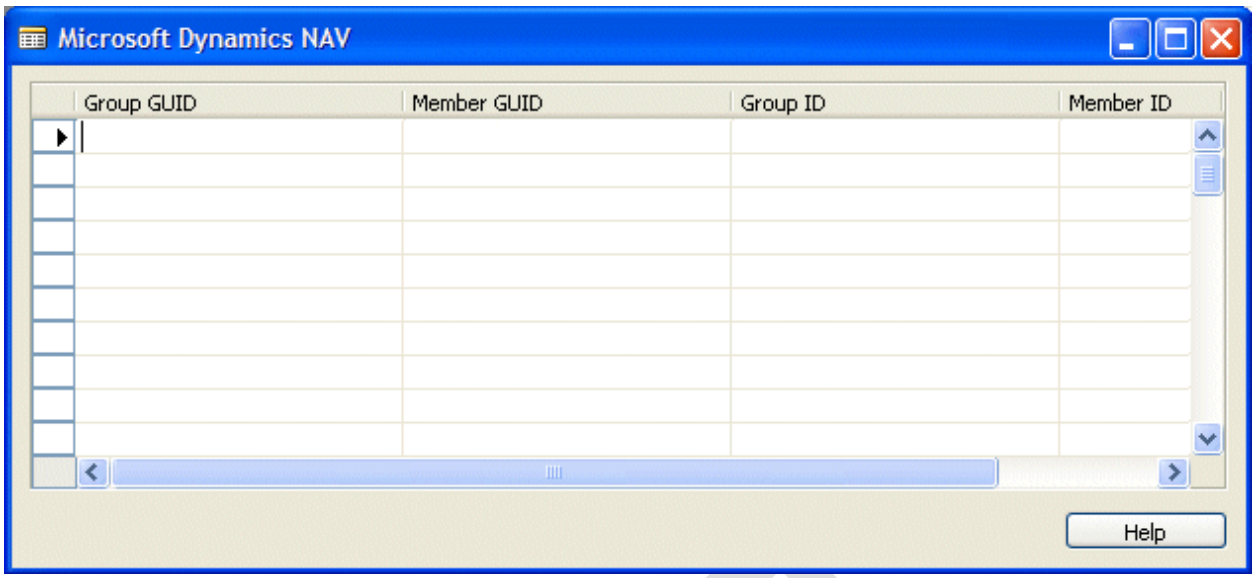


Виртуальная Таблица Windows Group Member (Члены Группы Windows)

В данной виртуальной таблице представлены члены Групп Windows, которые могут быть включены в систему безопасности Dynamics NAV. Члены групп Windows, у которых есть права доступа в системе безопасности Dynamics NAV не должных вводить пароль при подключении к Dynamics NAV. Данную информацию можно видеть, если клиент включен в Active Directory. Виртуальная таблица ***Windows Group Member*** содержит следующие поля:

Поле	Описание
Group GUID	Глобальный уникальный идентификатор (GUID) группы Windows
Member GUID	GUID члена группы Windows
Group ID	Идентификатор группы Windows.
Member ID	Идентификатор члена группы Windows

Чтобы получить доступ к виртуальной таблице ***Windows Group Member*** необходимо создать табличную форму:



Виртуальная Таблица SID - Account ID (SID – Account ID)

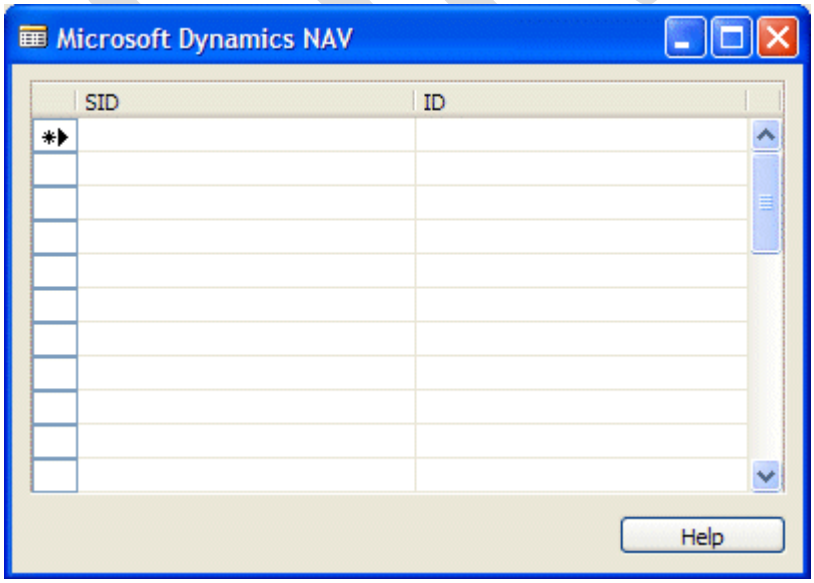
Данная виртуальная таблица может преобразовать уникальный идентификатор безопасности (SID) объекта Windows в идентификатор. Также она может преобразовать идентификатор объекта Windows в SID. Объектом Windows может быть пользователь Windows или группа. Виртуальная таблица ***SID - Account*** содержит следующие поля:

Поле	Описание
SID	Уникальный идентификатор безопасности (SID) пользователя или группы Windows
ID	Идентификатор пользователя Windows или группы Windows.

Примечание

Данная таблица всегда пуста.

Чтобы получить доступ к виртуальной таблице ***SID - Account ID*** необходимо создать табличную форму:



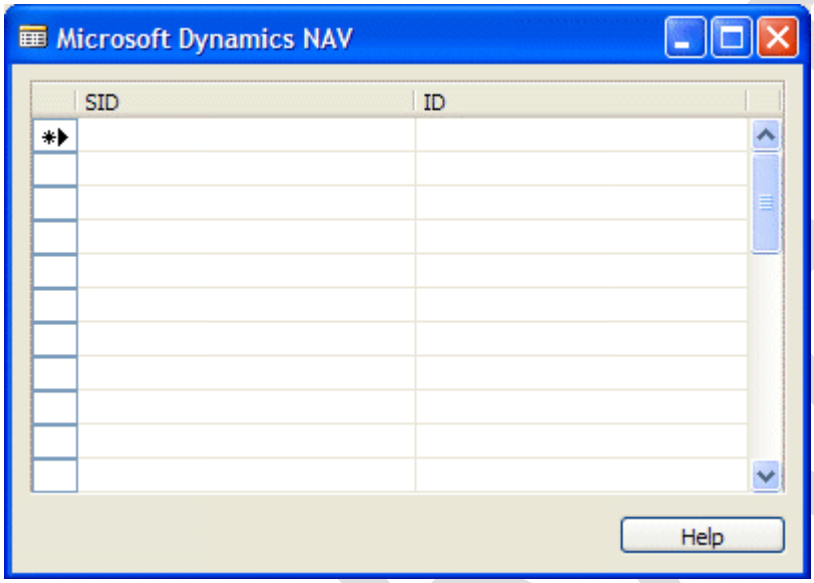
DRAFT

Виртуальная Таблица User SID (Пользователь SID)

В данной таблице отображаются уникальные идентификаторы безопасности (SID) и идентификаторы групп, члены которых вошли в систему. Виртуальная таблица ***User SID*** содержит следующие поля:

Поле	Описание
SID	Уникальный идентификатор безопасности (SID) группы, член которой вошел в систему.
ID	Идентификатор группы, член которой вошел в систему.

Чтобы получить доступ к виртуальной таблице ***User SID*** необходимо создать табличную форму:



DRAFT

ЧАСТЬ 4 ФОРМЫ

ГЛАВА 9 ФОРМЫ – БАЗОВЫЕ ПОНЯТИЯ

Формы используются для ввода и отображения данных. Например, можно использовать форму для ввода информации о новых клиентах или для обновления и просмотра данных об имеющихся клиентах.

В этой главе представлены базовые принципы и основные задачи, которые необходимо знать при конструировании и использовании форм.

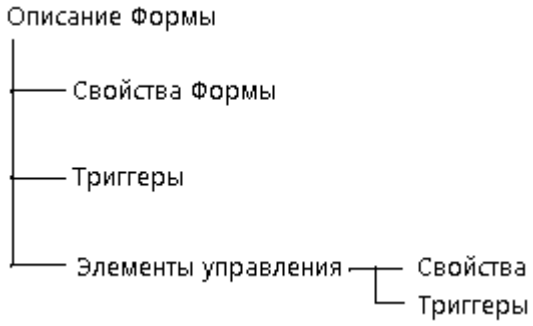
- Что такое формы?
- Создание форм
- Сохранение, компиляция и запуск форм

9.1 Что Такое Формы?

Следующим шагом в разработке приложения C/SIDE после создания таблиц должно стать конструирование форм. Формы используются для ввода информации в таблицы баз данных и для получения и отображения информации из этих таблиц. Именно через формы пользователи генерируют события, определяющие работу приложения.

Формы можно использовать для обращения к одной таблице или одновременно к нескольким различным таблицам с целью получения информации. Форма может отображать данные, которые вычисляются «на лету», в процессе отображения формы, или содержать информацию (например, надпись), которая не связана с какой-либо таблицей, либо просто декоративные элементы (например, растровые изображения).

На рисунке внизу показаны компоненты формы и связи между ними.



Формы создаются и редактируются в Form Designer.

DRAFT

Что Такое Элементы Управления?

Вся информация о форме представлена в элементах управления (controls). Элементы управления – это объекты, которые могут отображать данные из поля таблицы базы данных, значение C/AL-выражения, растровые изображения или статическую информацию, в частности, текст пояснительного характера.

Некоторые элементы управления называются контейнерными элементами управления (container controls). В качестве примера можно привести фрейм (frame). Фрейм сам по себе не отображает данные или информацию, но он может содержать ряд других элементов управления, которые требуется сгруппировать. Еще одним контейнерным элементом управления является закладка (tab control). Закладка – это, по сути, набор фреймов или страниц, которые помещены друг над другом. Пользователь может переключаться между страницами, щелкая по заголовкам закладок. Закладки позволяют группировать данные в форме так, чтобы каждая страница не была перегружена информацией, и можно было очень быстро и легко переключаться между страницами.

Еще один принцип – иерархия элементов управления, когда используются родительский и зависимые, или дочерние, элементы управления. Наилучший пример – текстовая зона (text box) и надпись (label). Дочерний элемент управления наследует некоторые свойства родителя, при этом всю иерархию можно перемещать на форме.

Что такое Связанные и Несвязанные Формы и Элементы Управления?

Как правило, форма связана с таблицей БД и используется для ввода данных в таблицу, и для отображения данных таблицы. Говорят, что такая форма связана с таблицей.

Несвязанная форма (unbound form) не относится к таблице. Примером несвязанной формы может служить форма, которая используется как меню, с помощью которого пользователь может выбирать и открывать другие формы или отчеты.

Элементы управления на форме, которая связана с таблицей, обычно связаны с полями этой же таблицы. Не обязательно для каждого поля в таблице устанавливать свой элемент управления, а все элементы управления на форме связывать с полями таблицы. Элементы управления, которые не связаны с полями, называются несвязанными элементами управления (unbound controls). Например, командная кнопка, которая позволяет распечатывать информацию, имеющуюся на форме или элемент управления, содержащий текст описательного характера. Важную категорию несвязанных элементов управления составляют элементы управления, отображающие вычисляемую информацию – будь то на основе таблицы или введенных пользователем значений. При создании, пустая форма является несвязанной.

Что такое Свойства Формы и Элемента Управления?

Свойства описывают, в том числе и то, как элемент управления размещен на форме, с каким полем он связан, и что происходит, когда в поле вводится информация. Различные типы элементов управления имеют различные наборы свойств. Например, текстовая зона – элемент управления, который обычно используется для отображения содержания поля базы данных. Он имеет больше свойств, чем зона изображения (picture box) – элемент управления, используемый для вывода на экран растровых изображений.

Сама форма тоже имеет свойства. Например, вы можете определить, должна ли форма использоваться только для отображения данных или разрешено вставлять новые и обновлять существующие записи. Свойства определяются в окне **Properties**, которое можно редактировать, когда форма открыта в Form Designer.

Что такое Триггеры?

Некоторые предопределенные события (events), которые происходят с формой или элементом управления, позволяют системе выполнять определяемую пользователем C/AL-функцию. Событие

DRAFT

активизирует функцию. Событие и функция в совокупности называются триггером (trigger). Триггеры форм включают функцию OnOpenForm, которая будет выполняться при открытии формы, и функцию OnModifyRecord, которая будут выполняться до того, как система примет изменения записи, введенные пользователем. Триггеры редактируются в окне C/AL Editor, которое можно открыть из Form Designer.

9.2 Создание Форм

Создавать формы можно вручную. Хотя этот метод предоставляет наибольшие возможности по контролю, на создание формы может потребоваться значительное время. Как вариант, для создания форм можно воспользоваться мастером форм (Form Wizard). Им достаточно просто пользоваться: потребуется указать необходимую для создания формы информацию, остальное матер форм сделает сам. После того как форма была создана мастером форм, ее можно изменить с помощью Form Designer так, чтобы она полностью удовлетворяла предъявляемым требованиям.

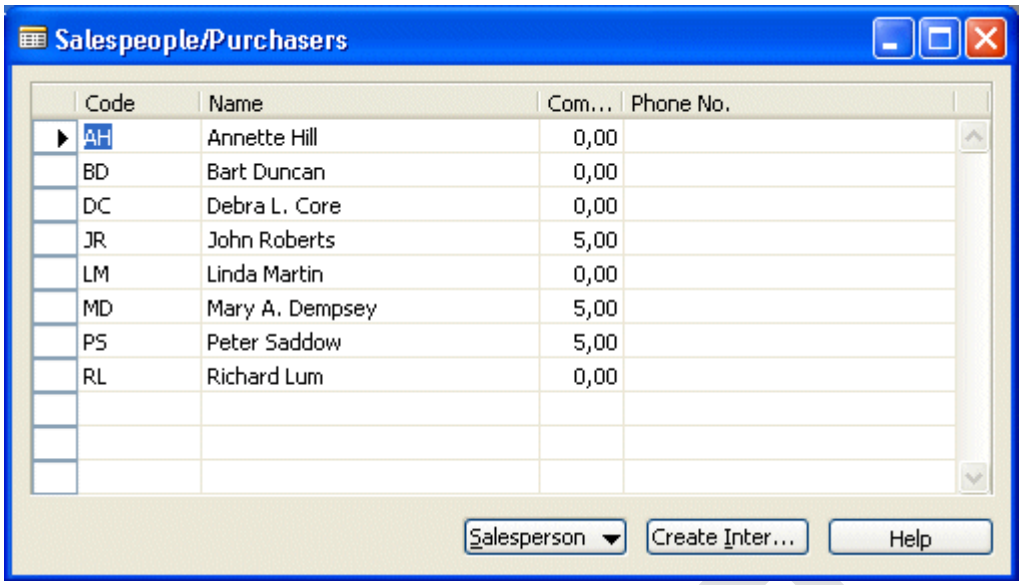
В Dynamics NAV можно создавать два вида форм:

- Карточная форма (Card Forms), которая показывает только одну запись.
- Табличная форма, которая сразу отображает несколько записей.

Оба вида форм можно создавать с помощью мастера форм.

Карточная форма (card form)

Табличная форма (tabular form)



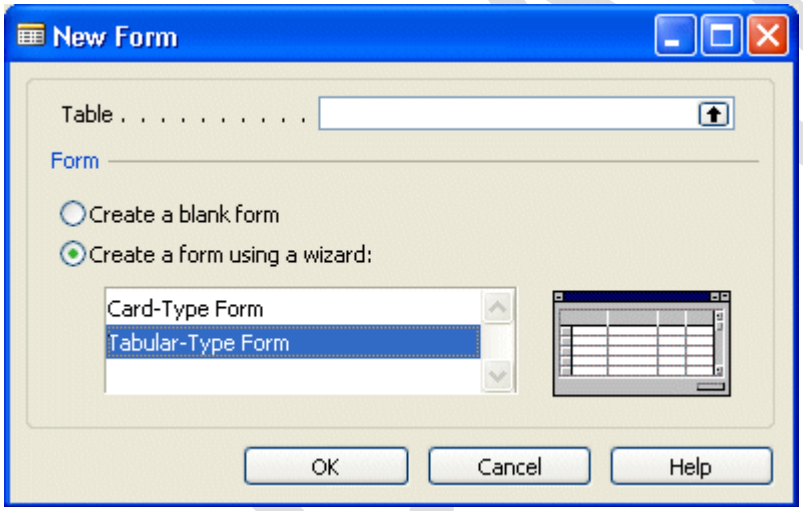
Создание Форм с Помощью Мастера Форм (Form Wizard)

Для создания форм с помощью мастера форм необходимо:

1. Щелкнуть Сервис, Object Designer (SHIFT+F12).

2. В **Object Designer**, щелкнуть кнопку **Form**, чтобы открыть список форм.

3. Щелкнуть кнопку **New**, чтобы открыть окно ***New Form***:



4. Если создаваемая форма связана с таблицей, в поле **Table** нужно ввести наименование таблицы или выбрать таблицу из списка.

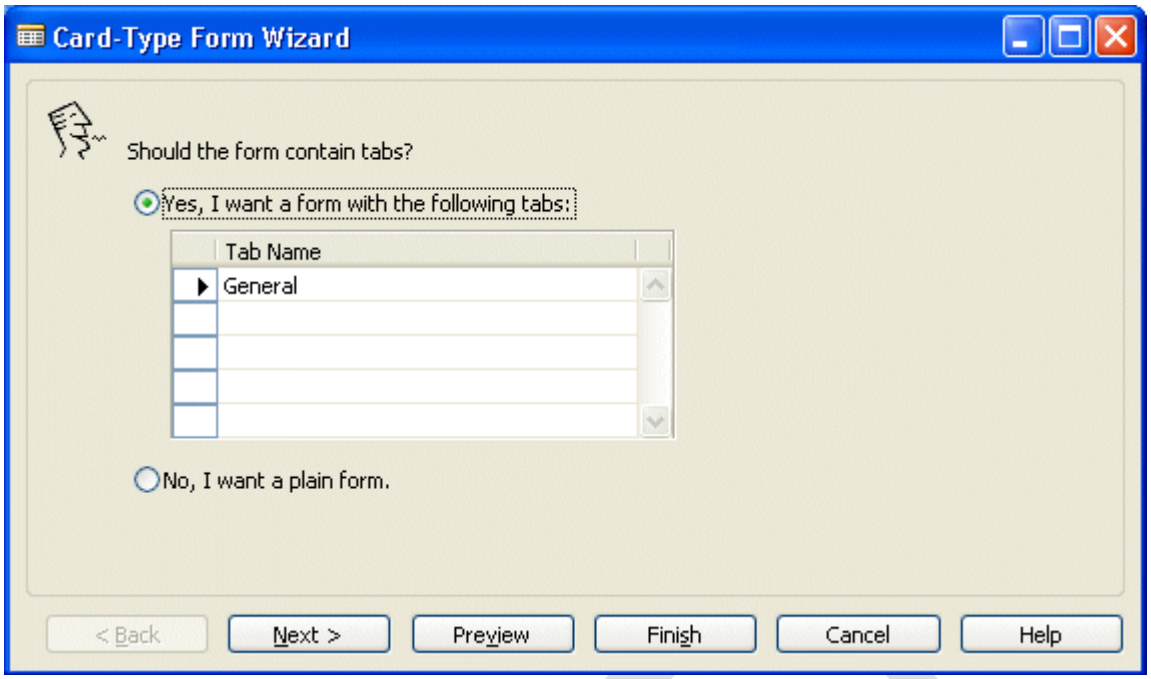
5. Выберите параметр "Create a form using a wizard" (Создать форму с помощью мастера).

6. Выберите вид формы, которую должен создать мастер: карточная форма или табличная форма. Щелкните кнопку OK.

Создание Карточной Формы

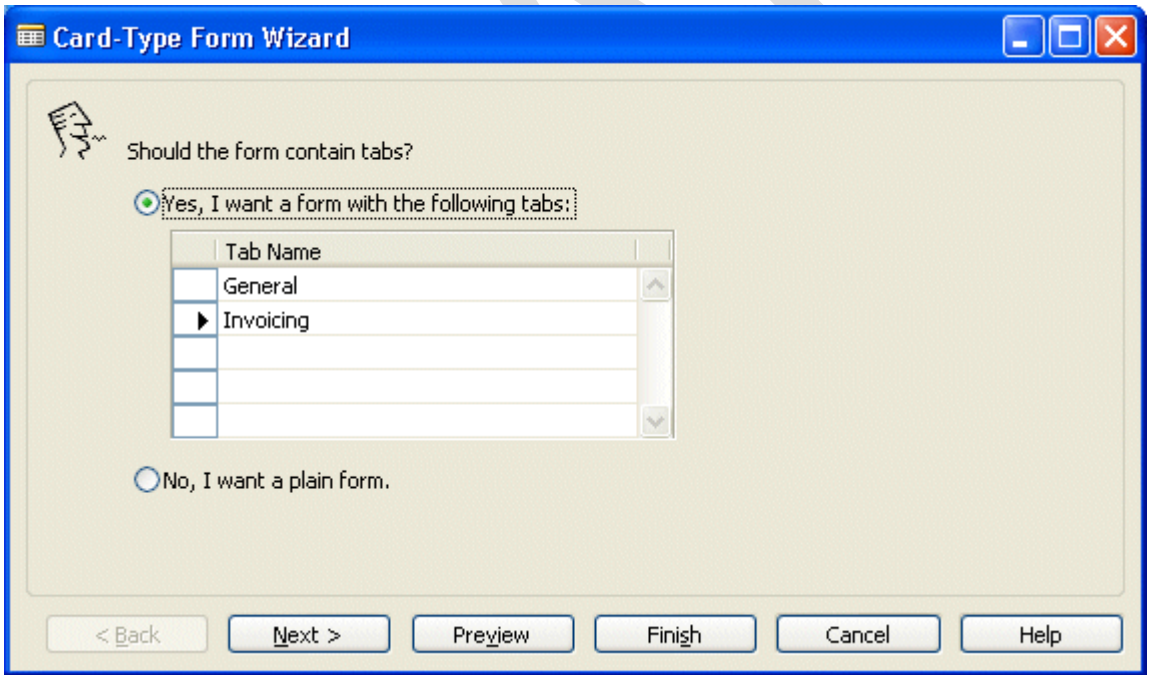
Для создания карточной формы, выполните шаги, приведенные ниже:

1. После того как вы указали, что мастер должен создать карточную форму и щелкнули кнопку OK, откроется Мастер Карточных Форм:



2. Во-первых, нужно решить будет ли форма содержать закладки. Форма, содержащая закладки – это многостраничная форма, которая позволяет переключаться между страницами, щелкая по закладкам.

В данном примере, будет создана форма с закладками, которая основана на таблице ***Customer*** (Клиенты).



3. Введите название каждой закладки и щелкните кнопку Next.

4. Далее необходимо решить какие поля из таблицы ***Customer*** требуется включить в форму.

Если вы создаете форму с закладками (tab control) вначале нужно выбрать закладку, на которой должны отображаться поля. Переключаться между закладками можно щелкая по ним мышью. Названия закладок будут такими, как вы определили на предыдущем шаге.

5. Мастер форм содержит два списка: **Available Fields,** в него включены все доступные поля таблицы, и **Field Order**, в который будут помещаться выбранные поля.

Чтобы вставить поле в список **Field Order** выберите требуемое поле в списке **Available Fields** и щелкните кнопку **>**. Чтобы вставить все поля сразу – нажмите на кнопку **>>**. Поле можно удалить из списка **Field Order,** для этого нужно нажать кнопку **<**. Чтобы удалить все поля нажмите кнопку **<<**.

> **Примечание**
>
> Содержимое списка **Available Fields** отображается согласно свойствам *Caption* полей таблицы, а не свойствам *Name*.

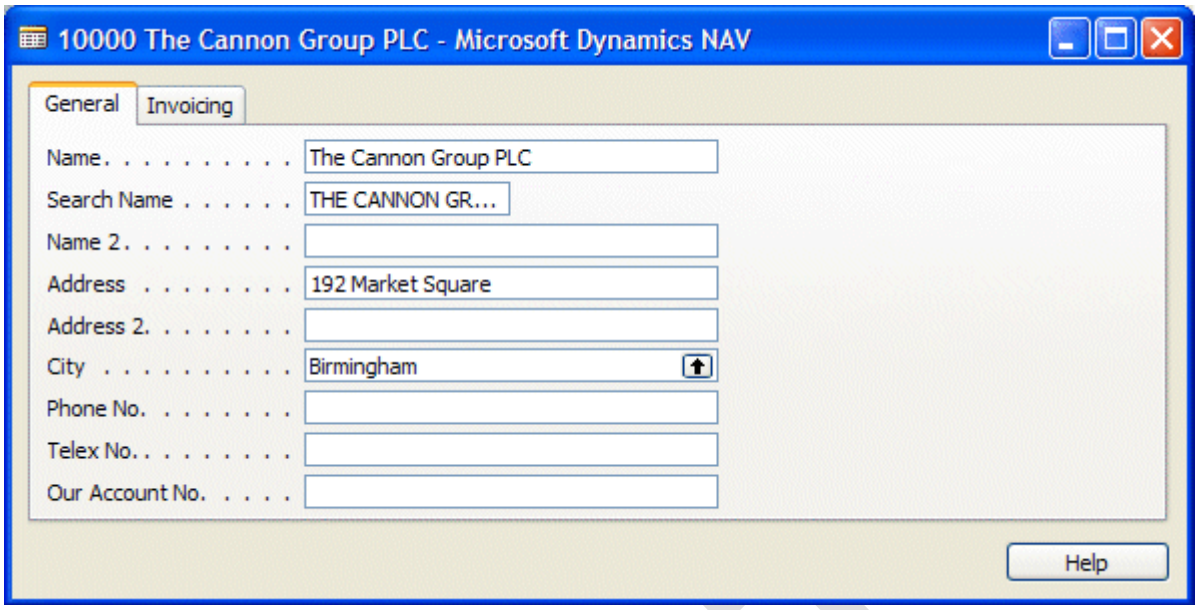
6. Поля будут добавлены на форму в таком же порядке, как они указаны в списке **Field Order.** Если требуется другой порядок, то нужно удалить поля из списка **Field Order** и вставить их в него снова, но уже в нужной последовательности.

7. Чтобы увеличить расстояние по вертикали между двумя элементами управления, щелкните кнопку **Separator**. Это позволит группировать информацию, улучшая, таким образом, визуальное восприятие. Помните, что лучше всего группировать информацию с помощью закладок. Разделитель (Separator) будет вставлен после выбранного в данный момент поля в списке **Field Order**. Разделитель можно удалить, для этого его нужно выбрать и щелкнуть кнопку **<**.

8. Чтобы вставить разрыв колонок щелкните кнопку **Column Break**. Правила вставки и удаления разрыва колонки такие же, как и для разделителя. Если вам нужно создать три и более колонки, лучше для этих целей использовать закладки.

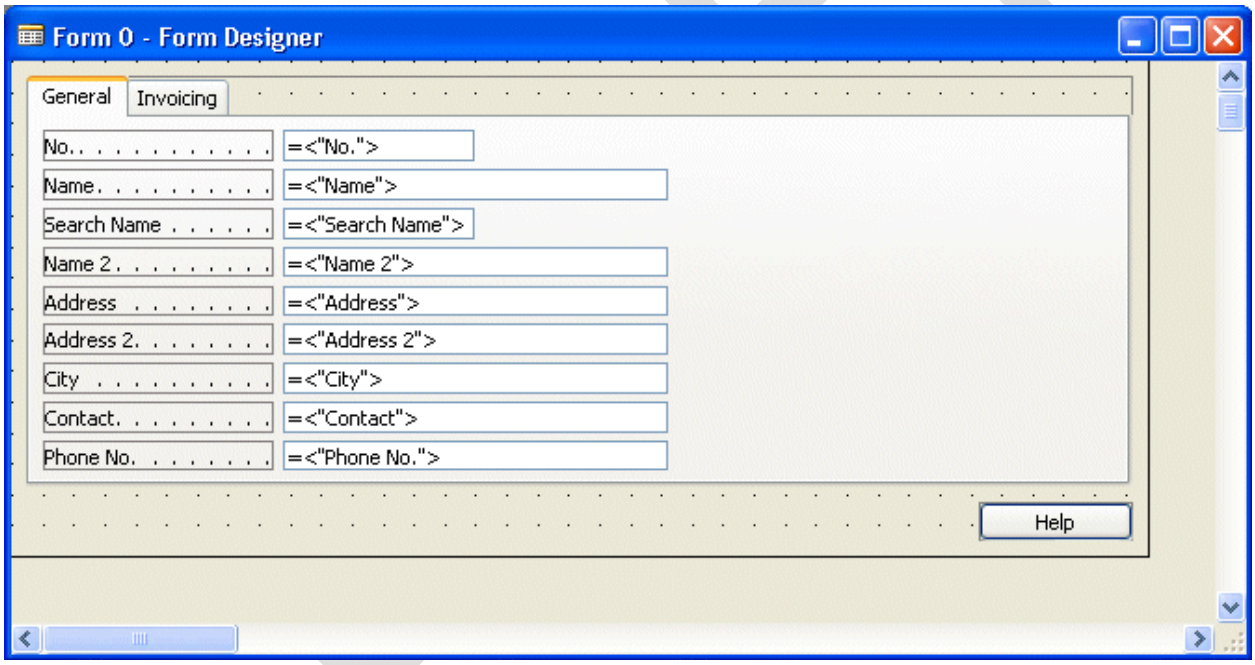
9. Чтобы посмотреть, как будет выглядеть форма, щелкните кнопку **Preview**.

DRAFT



10. Если вам не нравится, как выглядит форма просто закрой ее, и вы снова окажетесь в мастере форм. Теперь вы можете продолжать корректировать форму, пока она не будет вас удовлетворять.

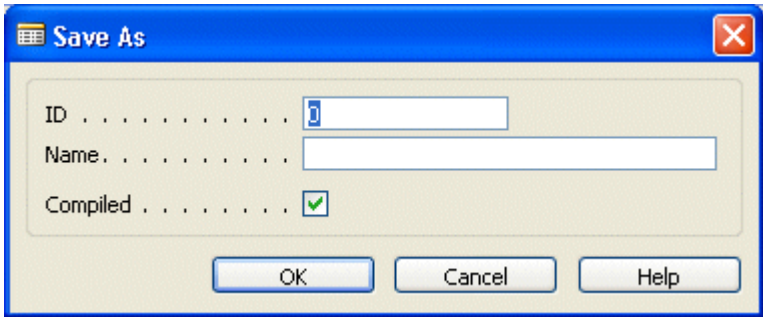
11. Нажмите кнопку **Finish**, чтобы завершить работу мастера, и открыть созданную форму в **Form Designer.**



Теперь вы можете вручную скорректировать форму (это будет рассмотрена позже)

12. Закройте форму. Система спросит, требуется ли сохранить форму.

13. Щелкните **Да**, появится окно ***Save As***:



14. Введите идентификатор и наименование новой формы, также можно указать, нужно ли компилировать форму.

Вы можете скомпилировать форму позже, выбрав ее в **Object Designer** и щелкнув Сервис, Компилировать.

15. После того как форма была сохранена и откомпилирована, ее можно запустить. Выберите форму в **Object Designer** и щелкните кнопку **Run**.

Создание Табличных Форм

Чтобы создать табличную форму выполните начальные шаги как описано в разделе «Создание Форм с Помощью Мастера». Далее выполните следующие шаги:

1. Необходимо решить какие поля из таблицы ***Customer*** требуется включить в форму.

Мастер форм содержит два списка: **Available Fields,** в него включены все доступные поля таблицы, и **Field Order**, в который будут помещаться выбранные поля.

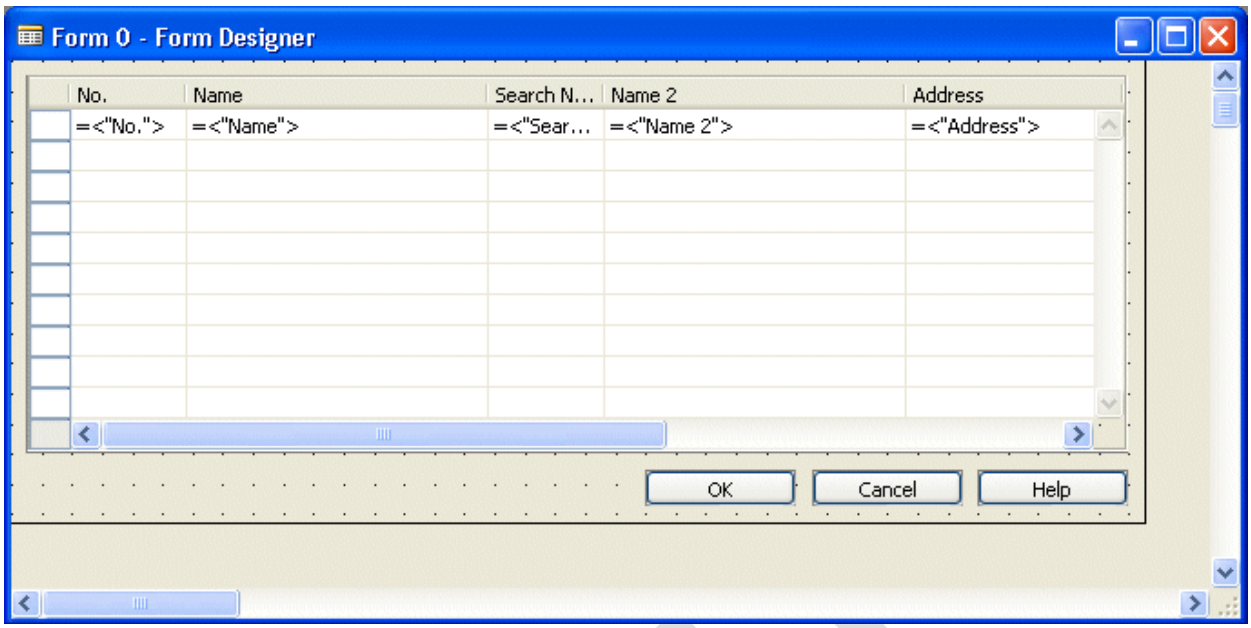
Чтобы вставить поле в список **Field Order** выберите требуемое поле в списке **Available Fields** и щелкните кнопку **>**. Чтобы вставить все поля сразу – нажмите на кнопку **>>**. Поле можно удалить из списка **Field Order,** для этого нужно нажать кнопку **<**. Чтобы удалить все поля нажмите кнопку **<<**.

> **Примечание**
>
> Содержимое списка **Available Fields** отображается согласно свойствам *Caption* полей таблицы, а не свойствам *Name*.

2. Поля будут добавлены на форму в таком же порядке, как они указаны в списке **Field Order.** Если требуется другой порядок, то нужно удалить поля из списка **Field Order** и вставить их в него снова, но уже в нужной последовательности.

3. Чтобы посмотреть, как будет выглядеть форма, щелкните кнопку **Preview**.

4. Нажмите кнопку **Finish**, чтобы завершить работу мастера, и открыть созданную форму в **Form Designer.**

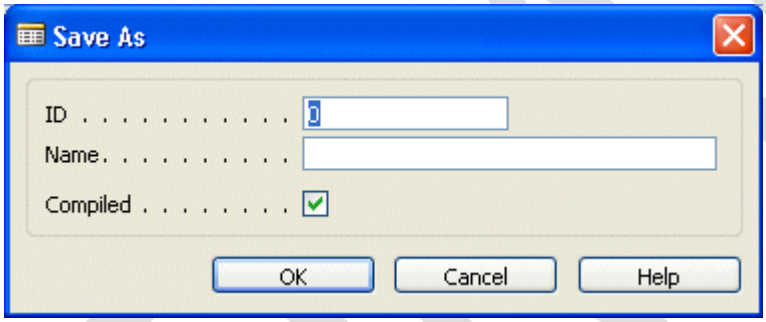


Теперь вы можете вручную скорректировать форму (это будет рассмотрена позже)

5. Закройте форму. Система спросит, требуется ли сохранить форму.

6. Щелкните **Да**, появится окно ***Save As***:

Вы можете скомпилировать форму позже, выбрав ее в **Object Designer** и щелкнув Сервис, Компилировать.



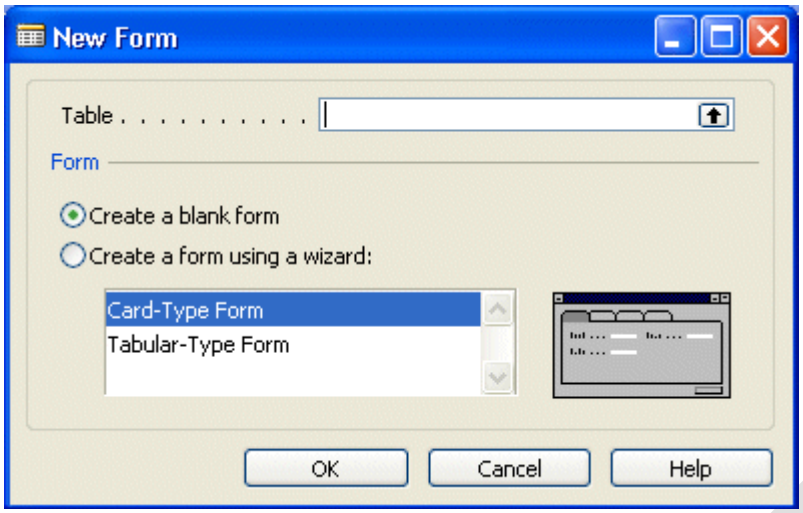
7. После того как форма была сохранена и откомпилирована, ее можно запустить. Выберите форму в **Object Designer** и щелкните кнопку **Run**.

Создание Формы без Использования Мастера

Чтобы создать форму без использования мастера:

1. Щелкните Сервис, Object Designer (SHIFT+F12).

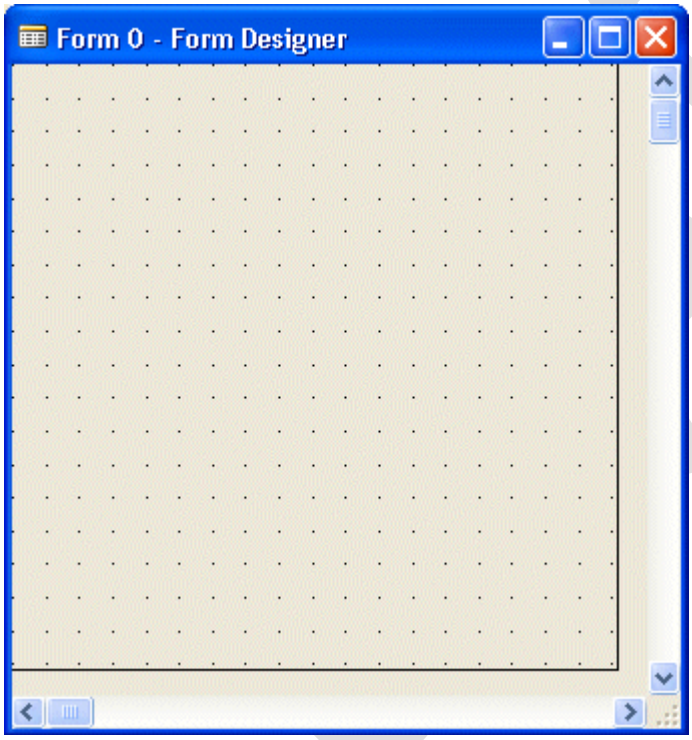
2. В **Object Designer**, щелкните кнопку **Form**, затем кнопку **New**. Откроется окно ***New Form***:



Если вы создаете форму, связанную с таблицей, введите название таблицы в поле **Table**. Также таблицу можно выбрать из списка (F6).

3. Выберите параметр «Create a blank form» (создать пустую форму) и щелкните кнопку OK.

Откроется **Form Designer**, отображающий пустую форму:



В главе 10 будет описано, как изменять форму, добавляя элементы управления, изменяя свойства и т.д.

4. Закройте **Form Designer** и щелкните **Да**, чтобы сохранить форму. Система предложит ввести идентификатор и наименование новой формы, также можно указать требуется ли компилировать форму при сохранении или нет. Вы можете скомпилировать форму позже, выбрав ее в **Object Designer** и щелкнув Сервис, Компилировать.

9.3 Сохранение, Компиляция и Запуск Форм

После того как структура формы разработана, форму нужно сохранить и скомпилировать, перед тем как ее можно будет запускать. Обычно это делается после окончания работ над формой. Однако вам может потребоваться сохранить форму, работа над которой еще не завершена, из-за

чего она не может быть скомпилирована (это бывает, если форма содержит C/AL код). Также вы можете выполнить текстовую компиляцию и тестовый запуск форму без ее закрытия и сохранения.

Сохранение и Закрытие Формы

Редактируемая форма закрывается, если закрыть Form Designer.

Чтобы сохранить форму:

1. При закрытии формы, C/SIDE спросит, требуется ли сохранить форму. Если это новая форма (форма которую до этого не сохраняли) потребуется задать для нее идентификатор и наименование. Идентификатор должен быть уникальным и соответствовать правилам нумерации объектов – ваш сертифицированный партнер Microsoft с компетенцией Business Solutions предоставит вам эту информацию.

Если вы ввели идентификатор (ID) и наименование (Name) в свойствах формы, система использует эти значения и их не потребуется вводить при закрытии формы.

2. Параметр **Compiled** выбран по умолчанию (установлен флаг). Если ваша форма не может быть скомпилирована, уберите флаг из данного поля.

3. Щелкните кнопку OK, чтобы сохранить форму.

Форму можно сохранить, не закрывая ее, для этого щелкните Файл, Save или Save As... Вы можете использовать Save As... чтобы дать форме новое наименование.

Компиляция Формы

Формы, как и другие объекты C/SIDE, должны быть скомпилированы, перед тем как их можно будет запускать. Как было описано ранее, компиляцию формы можно выполнять при сохранении.

В процессе работы над формой может потребоваться тестовая компиляция, она применяется для поиска возможных ошибок. Это особенно полезно, когда в триггерах формы содержится C/AL код. Тестовую компиляцию можно запустить, щелкнув Сервис, Компилировать.

Запуск Формы

В полностью завершенном приложении формы интегрированы в меню или вызываются из других форм. Однако при создании форм, часто необходимо запустить форму до того, как она будет интегрирована в приложение.

Вы можете запустить форму из списка форм в **Object Designer**, выбрав ее и щелкнув кнопку **Run**. Формы также можно запускать из **Form Designer**, щелкнув для этого Файл, Run.

DRAFT

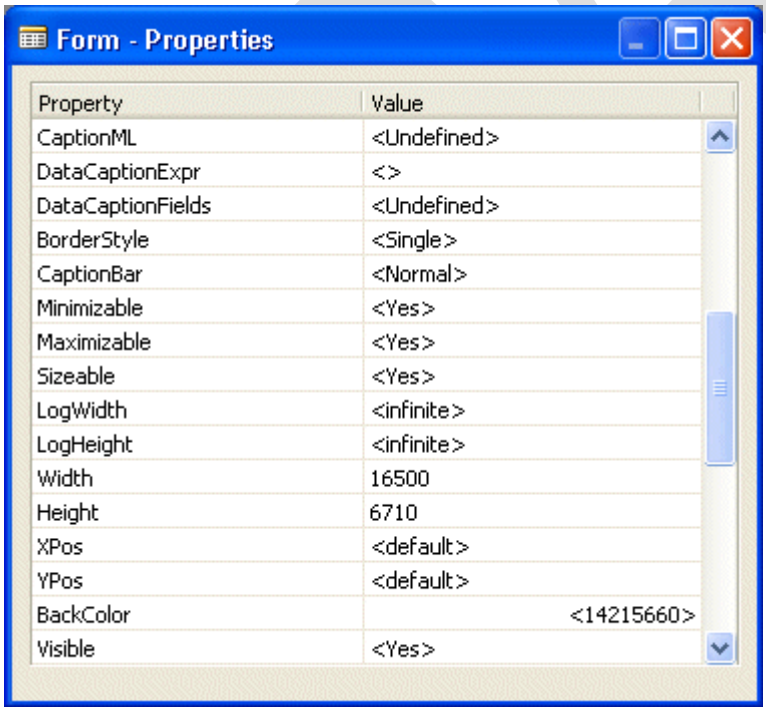
ГЛАВА 10 РЕДАКТИРОВАНИЕ ФОРМ

В данной главе будет описано, как редактировать формы, добавляя элементы управления и изменяя свойства форм и элементов управления.

- Свойства форм и элементов управления
- Типы элементов управления
- Добавление элементов управления
- Выбор, перемещение и изменение элементов управления
- Инструменты для настройки элементов управления
- Настройка свойств элемента управления
- Контейнерные элементы управления (Container Controls)
- Как использовать элементы управления в приложении

10.1 Свойства Форм и Элементов Управления

Настройка свойств может выполняться для всех объектов приложения. Редактирование свойств форм и элементов управления выполняется с помощью окна ***Properties,*** которое запускается из **Form Designer** (щелкните Вид, Properties или нажмите клавиши SHIFT+F4). Если выбрана форма или элемент управления, система отображает в окне ***Properties*** свойства выбранного объекта. В заголовке окна ***Properties*** показан вид, выбранного в данный момент, объекта (форма - form, текстовая зона - text box, надпись – label и т.д.). В первой строке окна отображается идентификатор объекта.



Каждая строка окна ***Properties*** содержит свойство, значение которого можно установить в поле **Value**. Значение свойства обновляется, когда вы покидаете поле или нажимаете клавишу ENTER. Если значение свойства содержит ошибку (например, для элемента управления установлен такой же идентификатор, как у другого элемента управления), значение обновлено не будет.

Значения по умолчанию отображаются в угловых скобках < >. Если для свойства есть значение по умолчанию, его можно восстановить. Для этого надо удалить текущее значение свойства и

покинуть поле. Некоторые свойства не имеют значений по умолчанию – обычно это свойства описывающие положение элемента управления на форме. Эти свойства обновляются C/SIDE при перемещении элемента управления.

Наследование Свойств

Элементы управления связанные с полями таблиц наследуют свойства, которые применяются к элементу управления. Эти свойства можно изменить, но нельзя переопределить свойства которые касаются проверки данных. Например, если в свойстве поля определено, что пользователь может вводить в него данные только в нижнем регистре, то с помощью свойств элемента управления нельзя разрешить ввод данных в верхнем регистре. Можно только сузить набор разрешенных символов, но нельзя расширить. С другой стороны можно, например, изменить свойство Caption, так как оно никак не связано с проверкой данных.

При разработке приложения вы должны решить, на каком уровне должны задаваться общие свойства – на уровне полей или на уровне элементов управления. Преимуществом использования низкого уровня (уровень поля) является гарантия целостности данных, так как если поле является источником данных для элемента управления, эти настройки применяются автоматически.

Свойства Формы

В следующей таблице приведено краткое описание некоторых наиболее важных свойств формы. Описания всех свойств приведено в онлайновой справке *C/SIDE Reference Guide*. Чтобы получить контекстную справку, откройте окно ***Properties*** для формы, поместите курсор на требуемое свойство и нажмите клавишу F1.

Название Свойства	Используется для…
ID	Числовой идентификатор объекта. Данное свойство может быть задано при сохранении формы. Идентификатор должен быть уникальным среди форм. Сертифицированный партнер Microsoft с компетенцией Business Solutions сообщит доступный вам диапазон номеров.
Name	Наименование формы. Не может быть двух форм с одинаковыми наименованиями
Minimizable	Можно ли сворачивать окно
Maximizable	Можно ли разворачивать окно.
Sizeable	Можно ли изменять размер окна
SavePosAndSize	Указывается должна ли сохраняться информация об изменении размера и положения формы, сделанных пользователем. Если в свойстве указано значение Да, то система сохранит изменения и при следующем открытии формы использует сохраненные настройки. Иначе будут использованы значения определенные при создании формы
Editable	Указывается можно ли редактировать элементы управления на форме. Если установлено значение Нет, элементы управления нельзя редактировать, даже если в свойстве Editable элемента управления установлено значение Да.
InsertAllowed	Можно ли использовать форму для вставки значений в базу данных
ModifyAllowed	Можно ли использовать форму для изменения записей в базе данных
DeleteAllowed	Можно ли использовать форму для удаления записей из базы данных

DRAFT

Название Свойства	Используется для…
CalcFields	Указывается список вычисляемых полей, вычисляемых системой при обновлении формы. Система автоматически рассчитывает значения вычисляемых полей, которые указаны в качестве источника для элемента управления, однако этого не происходит, если вычисляемое поле является, лишь частью выражения.
UpdateOnActivate	Должна ли система обновлять форму при активации
SourceTable	Указывается таблица-источник для форм. Обычно таблица указывается при создании формы. Если вы, создали форму, не указав таблицу, чтобы связать форму и таблицу можно указать название таблицы в этом поле.
SourceTableView	Создается вид (что можно видеть) для таблицы-источника данной формы. Здесь можно указать требуемый ключ, порядок сортировки и используемые системой фильтры.
SaveTableView	Указывается должна ли система сохранять информацию о записи, которая просматривалась перед закрытием формы, порядке сортировки и текущих фильтрах, а затем применять эту информацию при открытии формы.

Основные Свойства Элементов Управления

В следующей таблице приведено краткое описание общих свойств для элементов управления. Описания всех свойств приведено в онлайновой справке *C/SIDE Reference Guide*. Контекстную справку вы можете получить, открыв окно ***Properties*** для элемента управления и нажать F1, поместив курсор на конкретное свойство.

Название свойства	Используется для
ID	Числовой идентификатор элемента управления. Система присваивает очередной идентификатор автоматически. Идентификатор должен быть уникальным среди элементов управления и элементов меню (menu item), используемых на форме
Name	Название элемента управления
Caption	Указывается текст, который система будет показывать для этого элемента управления.
HorzGlue	Горизонтальная привязка элемента управления на форме. Можно выбрать значения: Left, Right или Both. Если указать Both, элемент управления будет изменять размеры при изменении размеров формы
VertGlue	Вертикальная привязка элемента управления на форме. Можно выбрать значения: Top, Bottom или Both. Если указать Both, элемент управления будет изменять размеры при изменении размеров формы.
Visible	Указывается, будет ли виден элементу управления при открытии формы. Данное свойство можно изменять из C/AL кода в процессе работы. Если элемент управления является дочерним, а для родительского элемента установлено свойство Visible = Нет, то элемент не будет виден, даже если установить ему свойство Visible = Да.
ParentControl	Указывается идентификатор родительского элемента управления.

DRAFT

10.2 Типы Элементов Управления

Данный раздел содержит краткое описание всех элементов управления, которые могут быть добавлены на форму. Список структурирован согласно категориям, к которым можно отнести элементы управления.

Статические Элементы Управления

Статические элементы управления (Static control) используются для отображения описательной или графической информации. Значения данных элементов управления нельзя изменять динамически.

К статическим элементам управления относятся:

Надпись (Label). Надпись используется для отображения текста, часто используется как заголовок другого элемента управления. В такой ситуации надпись обычно является дочерним элементом, однако надпись может использоваться на форме и отдельно.

Изображение (Image). Элемент управления Изображение используется для показа рисунков.

Фигура (Shape). Элемент управления Фигура используется для отображения графических элементов (линия, окружность, прямоугольник и т.д.).

Элементы Управления, Содержащие Данные

Элементы управления, содержащие данные (Data control), отображают значения выражений на языке C/AL, например, значения из полей таблиц или «настоящего» выражения (наименование поля таблицы является примером простейшего выражения). В следующей таблице приведены элементы управления и типы данных, для отображения которых они используются:

Элемент управления	Соответствующий тип данных
Check Box	Booleans и BLOB
Option Button	Все кроме BLOB
Text Box	Все
Picture Box	Boolean, option, integer и BLOB
Indicator	Integer, decimal, date, time

Элементов управления обязательно должны содержать ссылку на данные в свойстве *SourceExpr*.

Контейнеры

Контейнерные элементы управления (Container control) используются для группировки других элементов управления. Некоторые свойства контейнеров переопределяют свойства включенных в них элементов управления. Если контейнер нельзя редактировать (Editable = No), то нельзя редактировать ни один, содержащийся в нем элемент управления (даже если свойство Enabled этих элементов управления содержит значение Да).

Фрейм (Frame). Фрейм это обычный прямоугольник, в который могут быть помещены другие элементы управления. В Form Designer при перемещении фрейма можно также перемещать содержащиеся в нем элементы управления.

Закладки (Tab Control). Форма может быть представлена как книга, которая содержит несколько страниц или фреймов. В один момент времени отображается одна страница. Для перехода между страницами нужно щелкнуть по закладке.

Контейнеры, Содержащие Данные

Табличная Зона (Table Box). Это особенный вид контейнеров, который отображает повторяющиеся элементы управления, содержащие данные. Каждый элемент управления,

DRAFT

содержащий данные представлен отдельным столбцом табличной зоны, а в качестве заголовков столбцов используются статические элементы управления. Строки образуются многократным отображением всех элементов управления по вертикали. Если табличная зона используется для отображения записей таблицы, то каждая строка показывать одну запись.

Прочие Элементы Управления

Командная Кнопка (Command Button). Командная кнопка не связана с данными, она используется для выполнения действий. Действие выполняется если щелкнуть по кнопке мышью или нажать клавишу ENTER или Пробел, когда на кнопке установлен фокус.

Кнопка Меню (Menu Button). По кнопке меню можно щелкнуть также как и по командной кнопке, но при этом действие выполнено не будет. Вместо этого откроется меню, в котором содержатся доступные для выбора пункты меню (menu item).

Пункт меню (Menu Item). Это строка в меню, доступная для выбора. Пункт меню почти идентичен командной кнопке, т.к. при его выборе также выполняется действие.

Сопутствующая Форма (Subform). Сопутствующая форма используется для отображения второй формы в главной форме. Таким образом, главная форма может показывать данные из двух таблиц. Например, главная форма может быть карточной формой, которая показывает записи из таблицы клиентов, а сопутствующая форма - табличной формой, содержащей информацию обо всех сделанных данным клиентом покупках.

10.3 Добавление Элементов Управления

В данном разделе приведены несколько примеров, показывающих как добавлять на форму элементы управления, не используя мастера форм.

Toolbox

Для добавления элементов управления используете панель инструментов Toolbox. Чтобы открыть toolbox, щелкните Вид, Toolbox или нажмите кнопку Toolbox на панели инструментов. Щелкните по соответствующему значку на панели инструментов toolbox, чтобы выбрать нужный инструмент.

Pointer	
Надпись (Label)	Текстовая зона (Text Box)
Выключатель (Check Box)	Кнопка выбора (Option Button)
Командная кнопка (Command Button)	Кнопка меню (Menu Button)
Не используется	Фрейм (Frame)
Изображение (Image)	Зона изображения (Picture Box)
Фигуры (Shape)	Индикатор (Indicator)
Закладки (Tab Control)	Сопутствующая форма (Subform)
Табличная зона (Table Box)	Матричная форма (Matrix Box)
Не используется	
Lock	Add Label

DRAFT

Некоторые инструменты не включены в данную версию C/SIDE. Они недоступны, хотя значки уже присутствуют на панели инструментов.

Инструмент Pointer используется для выбора элементов управления на форме. При выборе других инструментов курсор изменит вид с «выбор» на «вставка».

Обычно, после того как вы вставили элементу управления на форму, требуется снова выбрать элемент управления на панели инструментов, перед тем как вставлять его на форму. Для вставки подряд нескольких элементов управления одного вида используйте инструмент Lock.

Если активизирован инструмент Add Label, со вставкой элементов управления вместе с ними будут добавляться элементы управления Надпись.

Добавление Текстовой Зоны (Text Box)

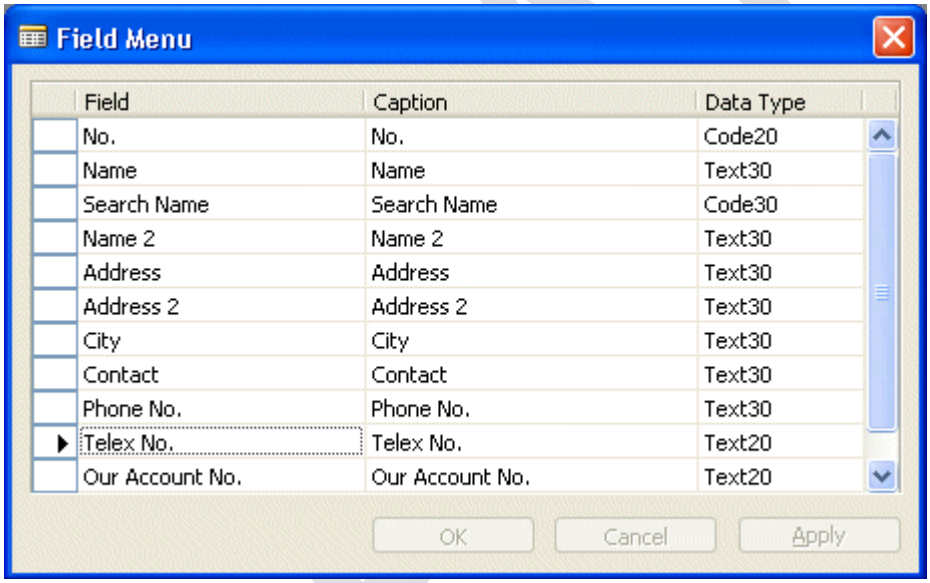
Если требуется добавить на форму текстовые зоны, связанные с полями таблицы базы данных, то самый простой способ это сделать – использовать Field Menu. Field Menu – это окно содержащее список полей таблицы, указанной в качестве источника для формы. Чтобы открыть Field Menu в Form Designer, нужно щелкнуть Вид, Field Menu.

Чтобы добавить текстовую зону, которая связана с определенным полем таблицы через свойство SourceExpr на форму:

1. Открыть Form Designer.

2. Создать пустую форму на основании таблицы ***Customer***.

3. Щелкнуть Вид, Field Menu, появится окно ***Field Menu***:



4. В окне ***Field Menu*** выбрать одно или несколько полей. В поле **Field** отображается свойство *Name,* а в поле **Caption** свойство *Caption*. Для получения дополнительной информации об этих свойствах, см. ГЛАВА 23 МНОГОЯЗЫКОВАЯ ФУНКЦИОНАЛЬНОСТЬ. В поле **Data Type** показывается тип данных поля.

5. Переместите курсор на редактируемую форму, он изменит вид на «вставка». Поля не нужно перетягивать, достаточно выбрать поля и переместить курсор на редактируемую форму.

6. В Form Designer щелкните один раз, чтобы активизировать дизайнер, а второй раз, чтобы вставить текстовые зоны. Текстовые зоны будут помещены под курсором. Позже их можно будет переместить.

Текстовые зоны создаются с размером по умолчанию. Чтобы вставить текстовые зоны другого размера, нужно щелкнуть и протянуть курсор.

При выборе более одного поля, текстовые зоны вставляются в один столбец под курсором. Они не связаны между собой и каждую из текстовых зон можно будет переместить.

Характеристики текстовой зоны:

- В свойства SourceExpr указана ссылка на поле таблицы.
- По умолчанию значение свойств *Name* и *Caption* такое же, как у поля таблицы, с которым связана текстовая зона.
- Большинство свойств текстовой зоны наследуются от поля таблицы, с которым связана текстовая зона.
- Если для текстовой зоны была вставлена надпись, то ее заголовок по умолчанию будет таким же, как заголовок поля таблицы (свойство Caption).

Преимуществом использования Field Menu является то, что при вставке текстовых зон и надписей, гарантируется соответствие названий и свойств.

Для полей всех типов данных, кроме Boolean создаются текстовые зоны, а для типов данных Boolean создаются выключатели (check box).

Добавление Текстовой Зоны без Использования Field Menu

Хотя проще всего текстовые зоны добавлять с помощью Field Menu, также текстовые зоны можно добавлять с помощью панели инструментов Toolbox. Этот способ применяется, когда нужно добавить текстовую зону, содержащую рассчитываемое значение. Также возможно создать не связанную текстовую зону, а связать ее с полем таблицы позже.

Чтобы добавить несвязанную текстовую зону:

1. Открыть форму в **Form Designer**.

2. Открыть панель инструментов **Toolbox** и выбрать инструмент **Text Box**.

3. Переместите курсор на редактируемую форму.

4. Щелкните, чтобы вставить текстовую зону с размером по умолчанию, или щелкните и протяните курсор, чтобы вставить текстовую зону другого размера.

Теперь на форме есть несвязанный элемент управления. Обратите внимания, что никакие характеристики не были унаследованы, а также для текстовой зоны не добавлена надпись.

Далее будет рассказано, как изменить свойства элемента управления, в том числе как связать текстовую зону с полем таблицы и добавить надпись.

Создание Надписей, Которые Показывают Описательный Текст

Надписи можно добавлять, даже если они не являются дочерними элементами для других элементов управления на форме. Обычно это делают, чтобы добавить описательный текст. Такой текст может содержать, например, инструкции по использованию формы или другую статичную информацию, не связанную ни с каким полем таблицы.

Чтобы добавить надпись:

1. Открыть форму в **Form Designer**.

2. Открыть панель инструментов **Toolbox** и выбрать инструмент **Label**.

3. Переместите курсор на редактируемую форму.

4. Щелкните, чтобы вставить надпись с размером по умолчанию, или щелкните и протяните курсор, чтобы вставить надпись другого размера.

DRAFT

Так как надпись не связана с другим элементом управления, для нее в свойствах Name и Caption устанавливаются значения по умолчанию (например, *Control4*). Свойства можно изменить в окне ***Properties***.

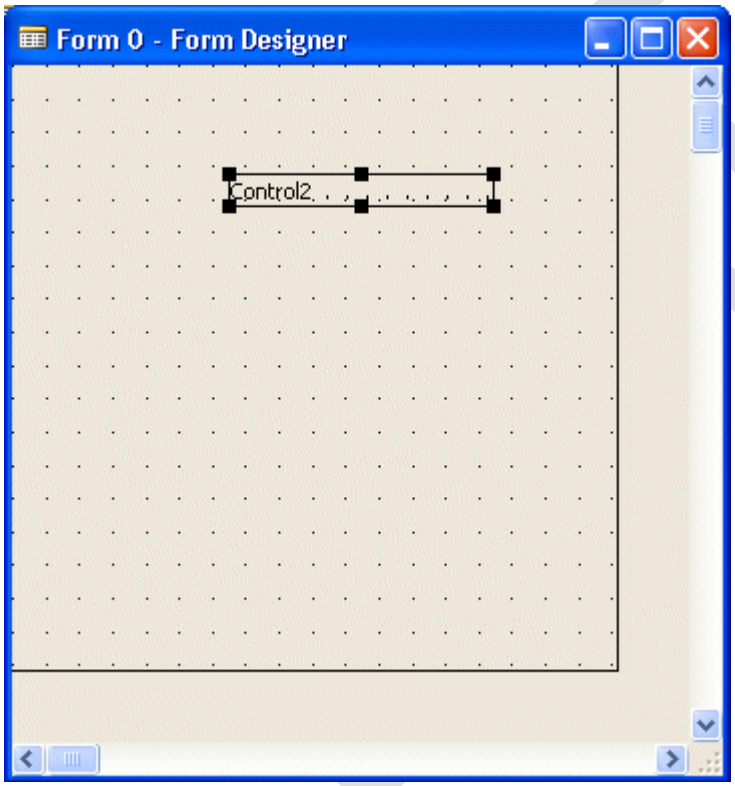
10.4 Выбор, Перемещение и Изменение Элементов Управления

В данном разделе будет описано, как выбирать и перемещать элементы управления, а также как выравнивать элементы управления и изменять их размеры.

Выбор Элемента Управления

Перед тем как перемещать, или изменять элемент управления, его нужно выбрать. Некоторые операции можно применять к одному элементу управления, другие к группе. Элементы управления можно выбирать по одному или группами.

Чтобы выбрать элемент управления, щелкните по нему мышью. При перемещении по редактируемой форме курсор изменяет вид, чтобы проще было определить, куда он указывает. По умолчанию курсор выглядит как крест, это означает, что курсор не указывает ни на один элемент управления. Если курсор указывает на элемент управления, он принимает вид на «выбор». Если щелкнуть по элементу управления вокруг него появляется рамка.

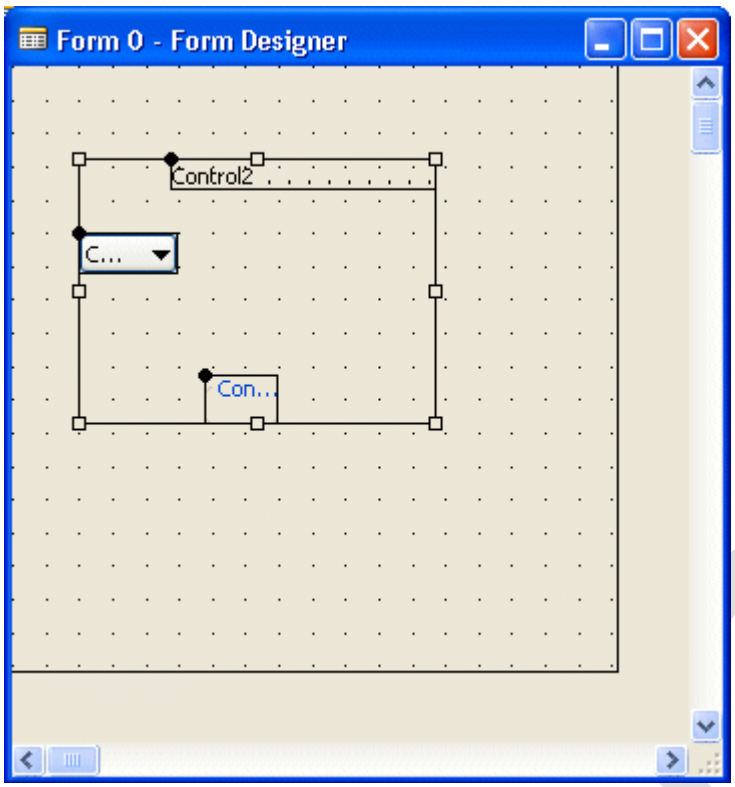


Множественный Выбор

Множественный выбор – это выбор нескольких элементов управления. Множественный выбор применяется, например, при выравнивании нескольких элементов (как выполняется выравнивание, будет описано позже).

Добавление к Выбранным Элементам

Если один элементу управления уже выбран и к выбору требуется добавить другие элементы управления нужно при выборе элементов управления удерживать клавишу CTRL. Если выбрано несколько элементов появляется рамка вокруг всех выбранных элементов, а в верхнем левом углу рамки каждого элемента управления появляется кружок.



Выделение Прямоугольником

Второй способ сделать множественный выбор – это использовать выделение прямоугольником. Когда курсор находится в пустом месте редактируемой формы и не указывает на элемент управления (выглядит как крест) нужно щелкнуть и удерживать кнопку мыши. Протяните курсор, появится прямоугольник.

Все элементы управления, которые полностью или частично попадают в прямоугольник, будут выбраны, а в верхнем левом углу рамки каждого элемента управления появляется кружок. Отпустите кнопку мыши, когда будут выбраны все требуемые элементы управления.

К выбранным элементам можно добавить еще элементы управления (как было описано выше). Также можно добавить элементы с помощью выделения прямоугольником (для этого надо удерживать нажатой клавишу CTRL).

> **Примечание**
>
> Для функции выделения прямоугольником доступна настройка, определяющая как должна работать данная функция. Щелкните Сервис, Настройка. В окне **Параметры** для опции **Выбор Элемента Полностью** можно указать значение Да или Нет. Если опция **Выбор Элемента Полностью** = Нет, выделение прямоугольником будет работать, как было описано выше. Если свойство **Выбор Элемента Полностью** = Да, элемент управления будет выбран, только если он попадает в прямоугольник полностью (не частично).

Выбор и Контейнерные Элементы Управления

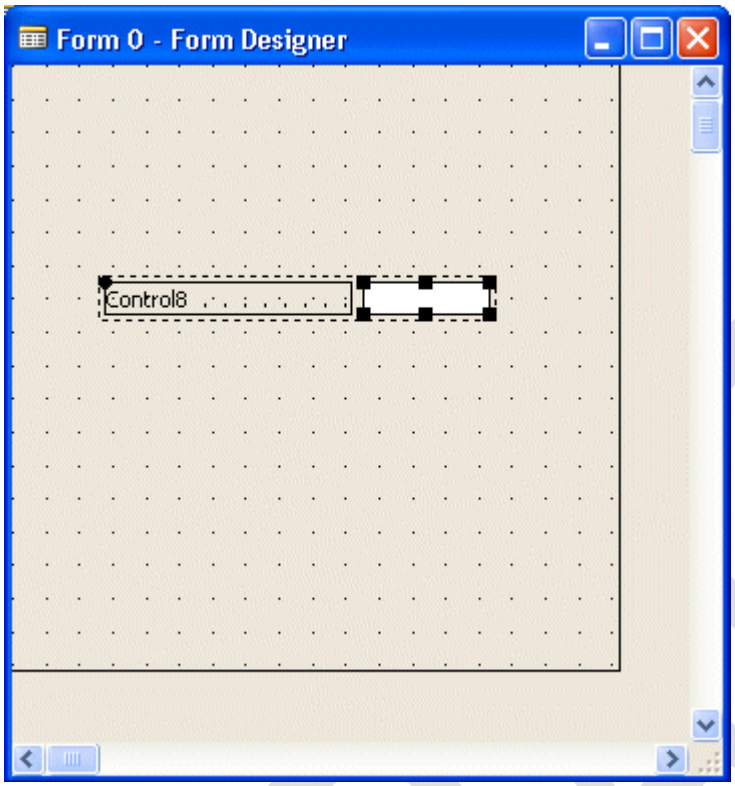
При выборе контейнерного элемента управления, все содержащиеся в нем элементы также будут выбраны, несмотря на то, что вокруг них не появится рамка. При перемещении контейнерного элемента управления, содержащиеся в нем элементы, так же будут перемещены.

Выбор и Связанные Элементы Управления

Связанные элементы управления состоят из самого элемента и одного или двух дочерних (подчиненных) элементов. Примером связанных элементов управления являются текстовые зоны с надписями. Надпись является дочерним элементом для текстовой зоны.

DRAFT

При выборе связанных элементов управления, таких как текстовая зона с надписью, вокруг элемента управления появится рамка. Вокруг дочернего элемента также появляется рамка с кружком в верхнем левом углу. Вокруг обоих связанных элементов появится общая пунктирная рамка. Если щелкнуть по рамке (курсор примет вид «выбор») дочерний элемент будет добавлен к выделению. Таким образом, будет получен множественный выбор, это позволит перемещать выделенные элементы одновременно.



Перемещение Элементов Управления

Когда вокруг элементов управления появилась рамка, за нее можно перемещать элементы. Для этого надо щелкнуть по рамке мышью и, удерживая кнопку нажатой, перетащить в нужное место. Затем нужно отпустить кнопку мыши. При перемещении нескольких элементов их порядок внутри выделения не изменяется.

Выравнивание Элементов Управления

Если форма была создана без использования мастера форм, или порядок некоторых элементов был изменен, возможно, потребуется выполнить точное выравнивание элементов.

Для простого и аккуратного выравнивания элементов управления в среде C/SIDE есть два метода:

1. Можно включить опцию Snap to Grid (щелкните Format, Snap to Grid). Теперь при перемещении элементов управления, они двигаются не плавно, а маленькими шажками. Точки на форме представляют некоторые точки координатной сетки, к которым привязываются элементы управления при перемещении.

> Совет: дистанция между точками координатной сетки задается в свойствах формы *HorzGrid* и *VertGrid*. Единица измерения – 1/100 мм.

2. Чтобы выровнять несколько элементов, нужно выделить несколько элементов управления, а затем щелкнуть Format, Align. В подменю выдерите один из четырех предложенных способов выравнивания элементов управления.

DRAFT

Например, если элементы управления выстроены в столбец и их требуется выровнять по вертикали по левому или по правому краю – выберите пункт Left или Right. Если для элементов управления выстроенных в столбец выбрать пункт меню Top или Bottom будут помещены друг на друга.

Изменение Размеров Элементов Управления

При добавлении элементов управления на форму с помощью мастера, им устанавливается размер по умолчанию. Иногда размер элемента управления нужно изменить, например, чтобы информация из большого поля таблицы отображалась полностью, или если для элемента управления был изменен размер шрифта.

Чтобы изменить размер элемента управления:

1. Выберите элемент управления. Вокруг, него появится рамка, на которой расположены восемь квадратов.

2. Выберите один из квадратов, курсор изменит свой вид на:

3. Щелкните левой кнопкой мыши и протяните курсор, чтобы установить новый размер элементу управления. Если включена опция Snap to Grid изменения размера элемента управления будет выполняться фиксированными шажками.

Изменение Размера Контейнерного Элемента Управления

Если создан контейнерный элемент, изменять размер каждого содержащегося в нем элемента нужно индивидуально. Размер контейнерного элемента, например, фрейма, изменяется также как и у других элементов управления. При изменении размера контейнерного элемента, содержащиеся в нем элементы не меняют ни своего размера, ни положения.

При увеличении размера контейнерного элемента, любой элемент управления, который при этом полностью окажется в внутри контейнера, будет автоматически преобразован в дочерний элемент данного контейнерного элемента.

Следует знать, что если при уменьшении контейнерного элемента, содержащийся в нем элемент управления оказался за пределами контейнера, то система все равно считает этот элемент частью контейнера. При этом данный элемент управления нельзя выбрать. Чтобы исправить это, нужно увеличить контейнерный элемент так, чтобы проблемный элемент управления оказался внутри контейнера.

10.5 Инструменты Настройки Элементов Управления

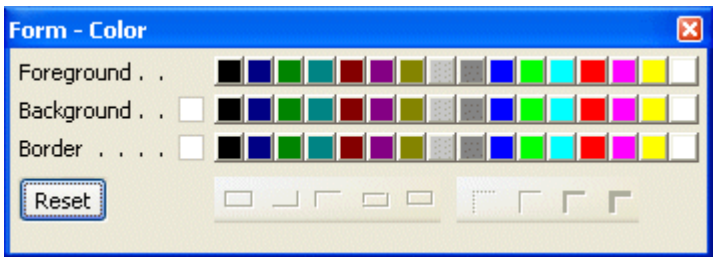
Кроме окна ***Properties*** для настройки некоторых свойств элементов управления используются специальные инструменты

- Инструмент Color, для настройки свойств, связанных с цветом, а также стилей границ.
- Инструмент Font, дня настройки свойств шрифтов.

Использование Инструмента Color

Чтобы открыть инструмент Color щелкните Вид, Color:

DRAFT



Для установки значений свойств *ForeColor*, *BackColor* и *BorderColor* нужно щелкать по цветным квадратам в соответствующих строках: foreground (текст), background (фон) и border (границы). Флажки в полях background (фон) и border (границы) соответствуют свойствам *BackTransparent* и *Border* (если в данных полях не установлены флажки, то выбранные цвета не оказывают никакого эффекта).

Если у элемента управления есть граница, то можно использовать девять кнопок в нижней части инструмента для выбора стиля и толщины границы.

Использование Инструмента Font

Чтобы открыть инструмент Font щелкните Вид, Font. Инструмент выглядит следующим образом:

Данный инструмент можно использовать для настройки свойств шрифтов выбранного элемента управления, который может отображать текст. Можно указать название шрифта, его размер, а также задать атрибуты (жирный, курсив и подчеркнутый), кроме того, можно задать горизонтальное выравнивание текста (left, center, right и general, *general* означается, что текст выравнивается по левому краю, а цифры по правому).

10.6 Настройка Свойств Элементов Управления

В данном разделе будет приведено несколько примеров демонстрирующих изменение свойств элементов управления, а также показаны ситуации, когда требуется изменение свойств.

Изменение Свойств Элементов Управления

Если элементы управления были добавлены без использования мастера форм, часто может потребоваться изменить некоторые свойства данных элементов. Изменение свойств элементов управления может потребоваться и в том случае, если они были добавлены на форму с помощью мастера, так как мастер не принимает во внимание все требования, предъявляемые к элементам управления.

Изменение Свойств Name и Caption Элемента Управления

У всех элементов управления есть свойства ID и Name. У элементов управления, содержащих данные, есть свойство *SourceExpr* (выражение-источник), которое содержит выражение на языке C/AL. Это может быть либо название поля в таблице или сложное выражение, возможно с использованием поля таблицы в качестве одного из элементов выражения.

При использовании мастера форм или инструмента Field Menu для создания текстовых зон, которые непосредственно связаны с полями таблиц, в свойства Name и Caption устанавливается значение по умолчанию из свойства Name поля таблицы. Если у поля таблицы задано свойство Caption, в этом случае для свойства Caption текстовой зоны будет использоваться данное значение. Для надписи (label) свойство Caption будет унаследовано от родительского элемента (текстовой зоны). Если

DRAFT

требуется более детальное описание, чем задано в свойстве Caption таблицы, то его можно поместить в соответствующее свойство обоих элементов управления.

Обратите внимание на следующие зависимости:

- Если изменить свойство *Caption* текстовой зоны, то значение свойства *Caption* надписи (label) отображается как значение по умолчанию (в угловых скобках). Если изменить значение свойства *Caption* текстовой зоны, то также будет изменено значение свойства *Caption* надписи.
- Если изменить значение свойства *Caption* непосредственно у надписи, то введенное значение будет отображаться без угловых скобок, т.е. значение больше не является значением по умолчанию. Если теперь изменить значение свойства у текстовой зоны, но значение свойства надписи останется без изменений.

Преобразование Несвязанного Элемента Управления в Связанный

Несвязанная текстовая зона или другой элемент управления, содержащий данные может быть легко преобразован в связанный элемент управления. Для этого надо просто изменить значение свойства SourceExpr.

Если в свойстве SourceExpr указать поле таблицы, значения свойств Name and Caption автоматически будут заполнены значениями по умолчанию. Автоматического добавления надписи к текстовой зоне не произойдет.

Если требуется изменить выражение-источник так, чтобы текстовая зона отображала данные из другой таблицы, то потребуется ввести более сложное выражение на языке C/AL.

Добавление Надписи к Текстовой Зоне

Если связанная текстовая зона, была создана преобразованием из несвязанной путем изменения значения свойства SourceExpr, то система не добавит надпись (label) автоматически. Надпись нужно добавить на форму с помощью панели инструментов Toolbox, а затем в свойстве *ParentControl* вместо значения по умолчанию (<Undefined>) указать идентификатор текстовой зоны. Идентификатор текстовой зоны можно посмотреть на первой строке окна ***Properties*** для текстовой зоны.

Свойства, Связанные с Отображением Данных

У элементов управления, как добавленных вручную, так и с помощью мастера, ряд свойств касающиеся отображения и форматирования данных заполняется значениями по умолчанию. Хотя это позволяет гарантировать единство дизайна по всему приложению, это может не соответствовать предъявляемым требованиям. Таким образом, может потребоваться изменить значения данных свойств, чтобы изменить способ отображения формы и ее элементов управления.

Управление Отображением Чисел

Далее приведено краткое описание свойств, определяющих способ отображения чисел. Полное описание данных свойств см. в онлайновой справке *C/SIDE Reference Guide*.

DecimalPlaces. Данное свойство (одновременно задаются минимальное и максимальное значения) определяет, сколько знаков после запятой должно отображаться, и сколько знаков после запятой можно ввести. Типичная ситуация, когда нужно использовать данное свойство это когда суммы хранятся в базе данных с пятью знаками после запятой (для точности), а на экране нужно отобразить, например 2 знака. В этом случае свойстве *DecimalPlaces* таблицы будет указано 2:5, а в свойстве текстовой зоны 2:2.

BlankNumbers. Из списка опций можно указать будет или нет отображаться ли определенный набор чисел.

BlankZero. Значение по умолчание Нет. Поля, содержащие значения 0 (ноль) или Нет, будут отображаться как пустые, если в данном свойстве установлено значение Да.

DRAFT

Divisor. Значение по умолчанию Undefined. При вводе числа, оно будет разделено на значение указанное в этом свойстве. Остаток от деления отбрасывается. Например, свойство *Divisor* можно использовать, чтобы показывать значения в тысячах, для этого нужно указать значение свойства *Divisor* =1000 (при этом оба числа 16400 и 16800 будут отображаться как 16).

Форматирование Отображаемых Данных

Далее приведено краткое описание свойств, определяющих способ форматирования данных. Полное описание данных свойств см. в онлайновой справке *C/SIDE Reference Guide*.

Format. Определяет, как система должна форматировать SourceExpr текстовой зоны.

Для каждого типа данных определены форматы по умолчанию. Также для выбора доступны наборы стандартных форматов. При необходимости можно создать свой собственный формат.

HorzAlign и VertAlign. Данные свойства определяют, как будут выравниваться по горизонтали и вертикали данные в текстовых зонах или заголовки в надписях.

MultiLine. В данном свойстве указывается, могут ли текстовые зоны или надписи содержать несколько строк текста. По умолчанию для данного свойства используется значение Нет, за единственным исключением: у надписей в заголовках столбцов табличной зоны (Table Box), данное свойство устанавливается Да

PadChar. В данном свойстве задается символ, который будет использован системой для заполнения пустого места строки. Символ может быть добавлен справа, слева или с обеих сторон одновременно, в зависимости от выравнивания текста, заданного в свойстве *HorzAlign*.

LeaderDots. В данном свойстве указывается должны ли появляться точки перед данными. Точки проставляются в зависимости от настроек выравнивания текста по горизонтали. Если данные прижаты влево, точки добавляются справа. Если данные прижаты вправо, точки добавляются слева. Если используется выравнивание по центру, точки добавляются и перед данными и после них. Если в свойство LeaderDots установлено значение Да, то оно переопределяет настройки свойства PadChar.

Свойства, Связанные с Контролем Ввода

Эта группа свойств используется для контроля значений, вводимых пользователем

Numeric. Если установить значение Да, то можно будет вводить только числовые значения.

MinValue, MaxValue. Устанавливаются минимальное и максимальное разрешенное значение.

ValuesAllowed. Указываются разрешенные для ввода значения. Значения разделяются точкой с запятой: *1;7;4711 или a;b;c*.

CharAllowed. Указываются разрешенные для ввода символы. Можно указывать диапазоны, например, *AZ* ограничивает ввод только латинскими символами в верхнем регистре, а *amot* указывает два диапазона: от a до m и от o до t.

NotBlank. Указывается, может ли значение быть пустым.

MaxLength. Указывает максимальное количество символов, разрешенных для ввода в текстовую зону.

AutoEnter. Если установить значение Да, система автоматически сохранит данные, когда в элемент управления будет введено максимально разрешенное количество символов и установит фокус на следующий элемент управления. Таким образом, при вводе не нужно будет нажимать клавишу ENTER.

PasswordText. Если установлено значение Да, вводимые значения не отображаются, вместо этого показываются звездочки ******.

DRAFT

Помощь Пользователю

Данная группа свойств используется для предоставления помощи пользователю.

ToolTip. Если в данном свойстве введен текст, то он будет отображаться в виде маленького всплывающего окна, которое появляется, если пользователь будет некоторое время удерживать курсор над данным полем.

DataCaptionField, DataCaptionExpr. Данные свойства используются, чтобы управлять надписями, отображаемыми в заголовке окна формы. Это статический заголовок, в котором обычно отображается название таблицы.

С помощью свойства DataCaptionField (либо на уровне таблицы, либо на уровне формы) можно указать поля, значение которых отображается в заголовке формы и обновляется при переходе по записям. В свойстве DataCaptionExpr (только для форм) можно указать выражение на языке C/AL, которое будет отображаться в заголовке формы. Выражение перерассчитывается при переходе к следующей записи.

> **Примечание**
>
> Если строка состояния включена (щелкните Сервис, Настройки, Панель Статуса = Да), то в строке состояния будет отображаться заголовок активной текстовой зоны или выключателя, а также текущее содержанием элемента управления.

10.7 Контейнерные Элементы Управления

Как было рассказано ранее, контейнеры используются для объединения нескольких элементов управления в логические группа. Это позволяет сделать приложение простым в использовании и интуитивно понятным для пользователя. Это также может упростить разработку и поддержку.

Использования Фреймов (Frame)

Фрейм используется для объединения других элементов управления. Когда элементы управления находятся во фрейме, некоторые операции, выполняемые над фреймом, также оказывают влияние на них влияние. Например, элементы управления перемещаются вместе с фреймом, при этом их положение внутри фрейма не изменяется, или если фрейм сделать невидимым, то невидимыми станут и все элементы управления, содержащиеся в нем.

У фрейма есть граница, которая может быть приподнятой или утопленной. Это позволяет визуально выделить группу элементов управления (например, кнопки выбора).

Чтобы создать фрейм, содержащий элементы управления:

1. Открыть форму в Form Designer.

2. Выбрать инструмент Frame, затем щелкнуть и протянуть курсор в редактируемой форме для создания фрейма.

3. Создать обычным способом элементы управления и поместить их во фрейм.

4. Установите свойства фрейма так, чтобы он соответствовал вашим целям.

Обычно изменяют свойство *ShowCaption,* чтобы у фрейма не отображался заголовок. По умолчанию, стиль границ фрейма Raised. Если назначение фрейма заключается не в выделении элементов управления, то чтобы границы не отображались, потребуется установить в свойство *Border* значение *No*.

Если существующие элементы управления перемещены во фрейм, элемент будет включен в состав фрейма. Если перенести элемент за пределы фрейма, он больше не будет принадлежать фрейму. При удалении фрейма, все содержащиеся в нем элементы управления также удаляются.

DRAFT

Создание Закладок (Tab Control)

При создании форм, базирующихся на таблице с большим количеством полей, удобно использовать закладки. Вместо создания больших форм переполненных элементами управления, можно сгруппировать элементы управления на страницах. Пользователь, щелкая по закладкам, выбирает, какие элементы управления должны отображаться.

Закладки можно создавать с помощью мастера форм.

Чтобы создать закладки вручную:

1. Откройте форму в **Form Designer**.

2. Выберите инструмент **Tab Control**, затем щелкните и протяните курсор в редактируемой форме.

3. Откройте окно ***Properties*** закладки и создайте требуемые страницы. Для этого нужно ввести названия страниц, разделив их точками с запятыми в свойство *PageNames*. Названия страниц используются в качестве заголовков закладок. Выбор страницы выполняется щелчком мыши по соответствующей закладке.

Далее нужно добавить на страницы элементы управления. Каждую страницу можно рассматривать как фрейм и работать с ней как с фреймом.

Создание Табличной Зоны (Table Box)

Табличная зона используется, когда нужно одновременно вывести много записей из базы данных. Табличная зона состоит из столбцов и строк. Между записями можно перемещаться, используя вертикальную полосу прокрутки, клавиши управления курсором или PAGEUP и PAGEDOWN. Если форма слишком узкая для отображения всех столбцов, автоматически появляется горизонтальная полоса прокрутки.

Табличную зону можно создать с помощью мастера форм.

Чтобы создать табличную зону вручную:

1. Откройте форму в **Form Designer**.

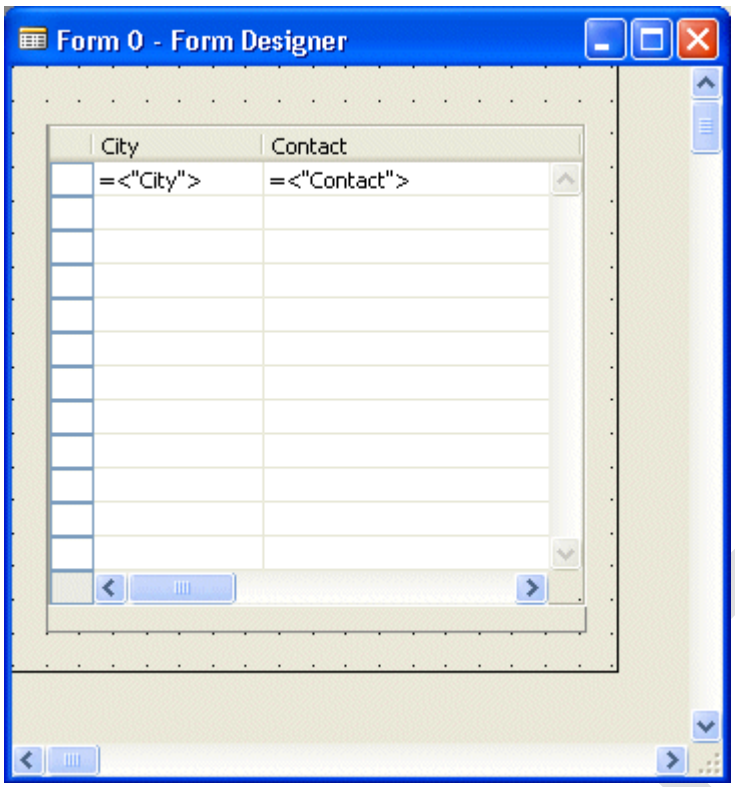
2. Выберите инструмент **Table Box**, затем щелкните и протяните курсор в редактируемой форме.

3. Если форма не связана с таблицей, то можно установить связь, указав в свойстве *SourceTable* название таблицы.

4. Щелкните Вид, Field Menu, чтобы открыть окно ***Field Menu***.

5. Выберите в окне **Field Menu** поля, которые нужно вставить в табличную зону, а затем дважды щелкнуть по табличной зоне. Первый щелчок активизирует **Form Designer**, а второй вставит поля. В столбцах будут отображены поля, а в строках записи из таблицы. В качестве заголовков столбцов система добавит элементы управления надпись.

DRAFT



10.8 Как Использовать Элементы Управления в Приложениях

В данном разделе описано несколько приемов, используемых для управления отображением данных в приложениях.

Отображение Нескольких Строк Текста

Для отображения длинных полей таблицы, использование однострочных текстовых зон, является не лучшим вариантом. Вместо этого можно настроить текстовую зону таким образом, чтобы текст разбивался на несколько строк.

Для создания многострочной текстовой зоны:

1. Открыть форму в **Form Designer**.

2. Выбрать нужную текстовую зону и увеличить ее по вертикали.

3. Открыть окно ***Properties*** для данной текстовой зоны (SHIFT+F4) и установить для свойства *MultiLine* значение Да.

4. Запустите форму. Ввод и редактирование текста выполняется по прежнему, в одну строку, которая прокручивается по горизонтали. Когда фокус покидает данную текстовую зону, ее содержание будет преобразовано в несколько строк. Разрывы строк автоматически вставляются после символов «Пробел». Пользователь может вставить принудительные разрывы строки, для этого нужно вставить символ «\» (если нужно отображать значение «\», то введите «\\»).

Отображение Рассчитанных Значений

Элементы управления могут использоваться также для отображения значений рассчитанных при отображении формы, а не хранящихся в базе данных

Чтобы отобразить рассчитанные значения:

1. Открыть форму в Form Designer.

DRAFT

2. Выбрать на панели инструментов **ToolBox** инструмент, соответствующий добавляемому элементу управления (выключатель, текстовая зона, индикатор).

3. Дважды щелкните в редактируемой форме, чтобы добавить элемент управления.

4. Откройте окно ***Properties*** для элемента управления. Введите в свойстве *SourceExpr* выражение.

Пример

Предположим, что существует таблица, содержащая поля Unit Price (Цена Единицы) и Employee Discount Rate (Скидка для Сотрудников %). На форме требуется отобразить цену для сотрудника с учетом скидки. Для этого надо добавить текстовую зону и ввести в свойство SourceExpr:

"Unit Price" - ("Unit Price" * "Employee Discount Rate" / 100)

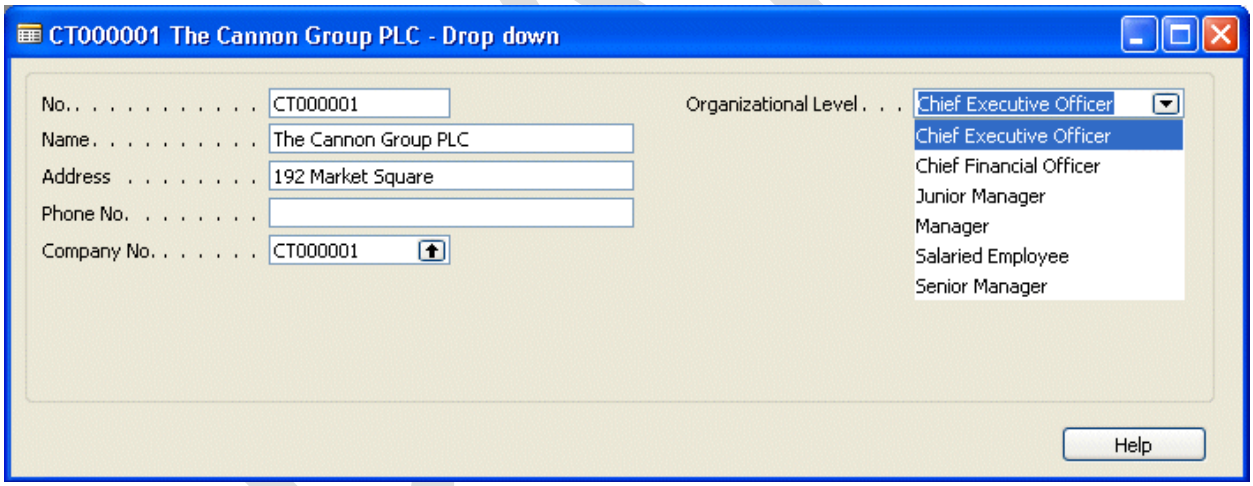
Представление Набора Опций

Одной из частых задач в программировании приложений является предоставление пользователю возможности выбора на форме одного значения из предопределенного набора. Например, в приложении, где пользователь часто должен вводить должность, система может содержать список должностей.

В среде C/SIDE набор опций можно представить несколькими способами. В следующих разделах будет описано два различных подхода.

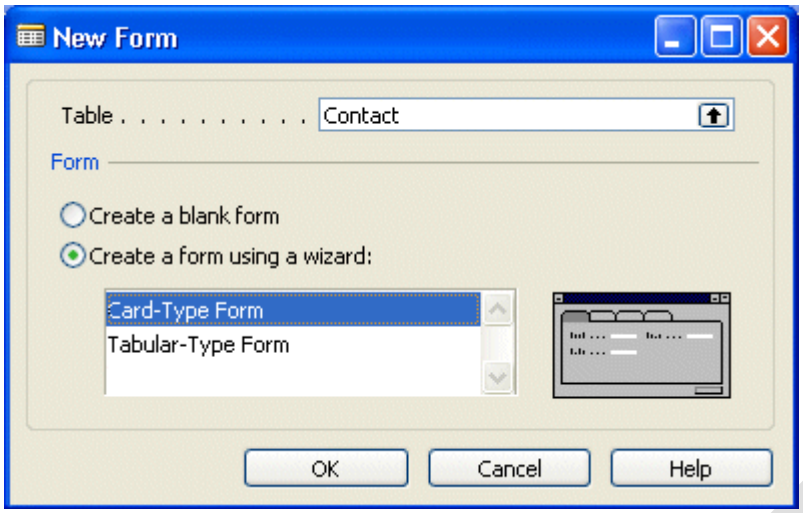
Создание Выпадающего Списка Опций

Список опций может быть представлен в виде выпадающего списка:

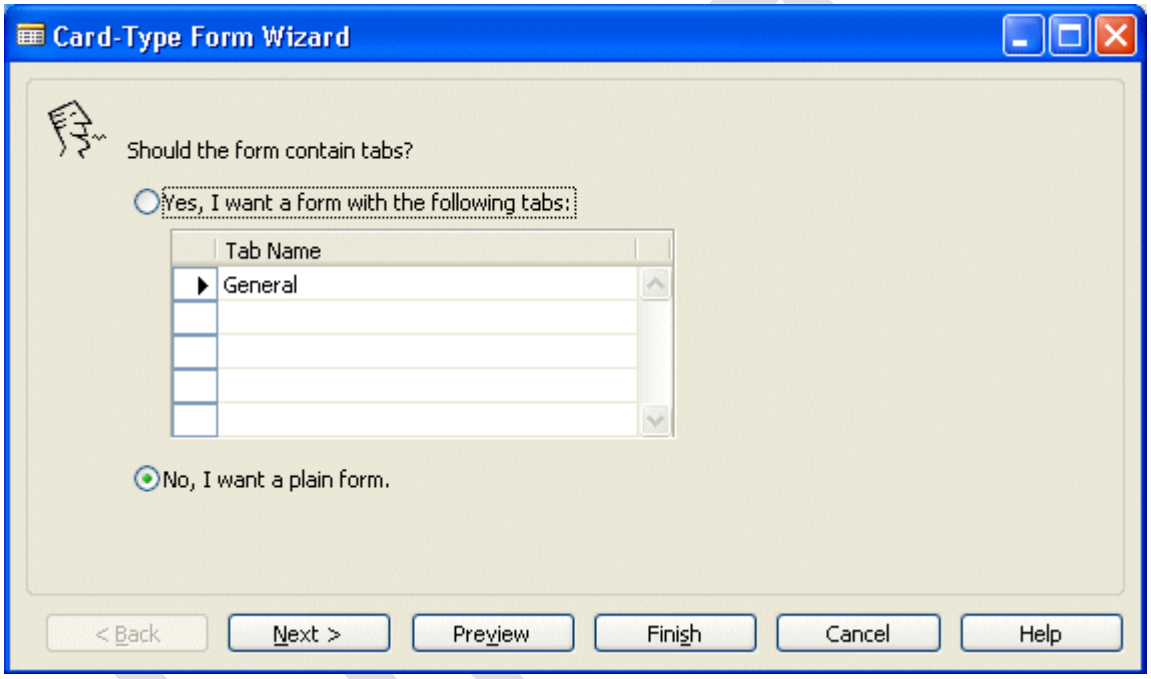


Перед созданием выпадающего списка, нужно создать форму.

1. Открыть **Object Designer**, щелкнуть кнопку **Form**, затем **New** для открытия окна ***New Form***:

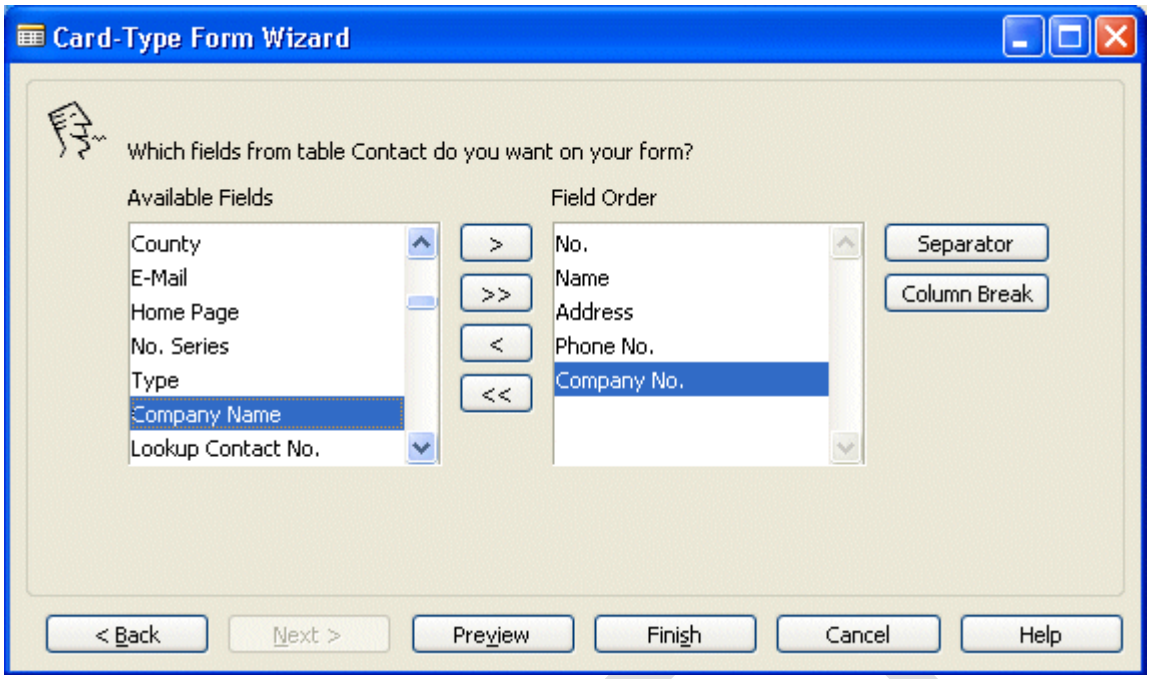


2. Создать карточную форму на основании таблицы ***Contact*** (Контакты).

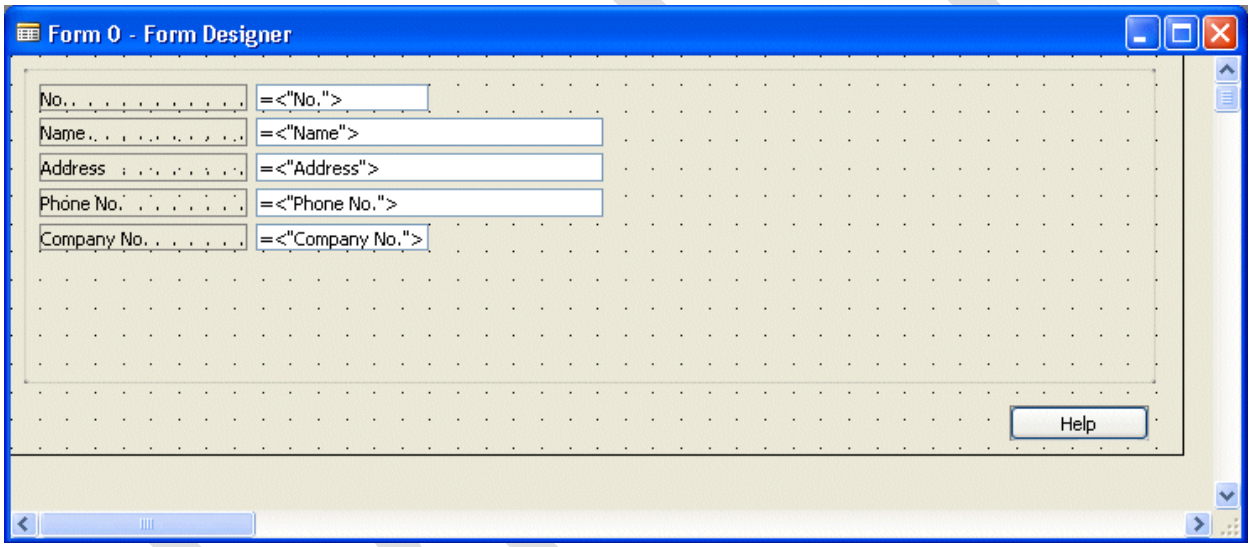


3. Выберите опцию «No, I want a plain form» в окне ***Card-Type Form Wizard*** и щелкните кнопку Next.

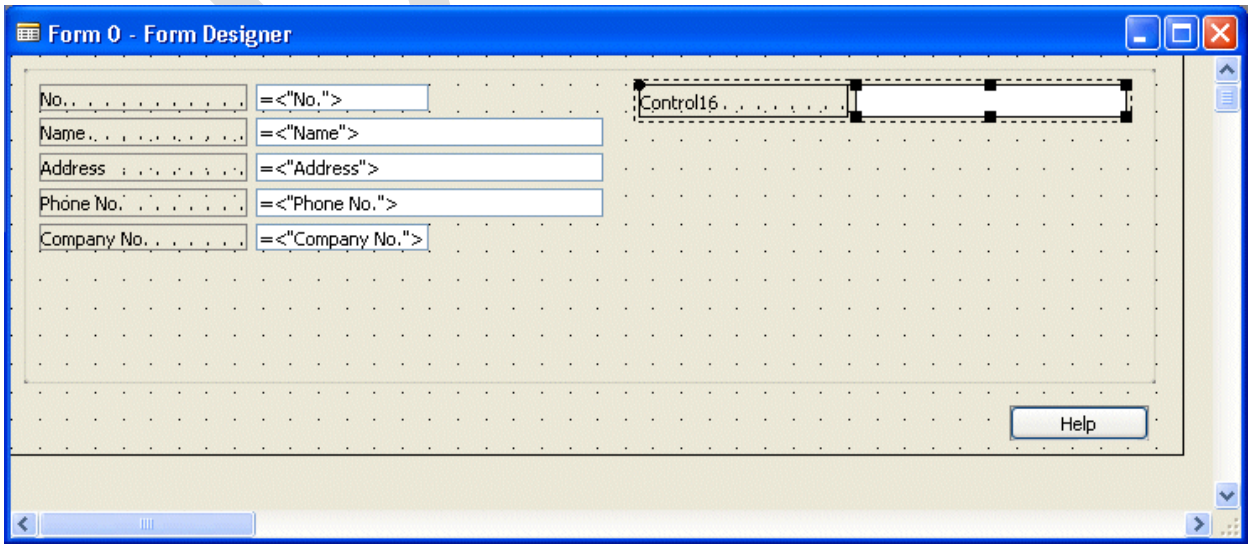
4. Добавьте на форму следующие поля:



5. В окне ***Card-Type Form Wizard*** щелкните кнопку **Finish**.



6. Вставьте текстовую зону и надпись:



Базовая форма создана. Следующим шагом будет создание выпадающего списка опций.

1. Выберите созданную текстовую зону и откройте окно ***Properties*** (SHIFT+F4).

2. В свойстве **Caption** введите *Organizational Level*.

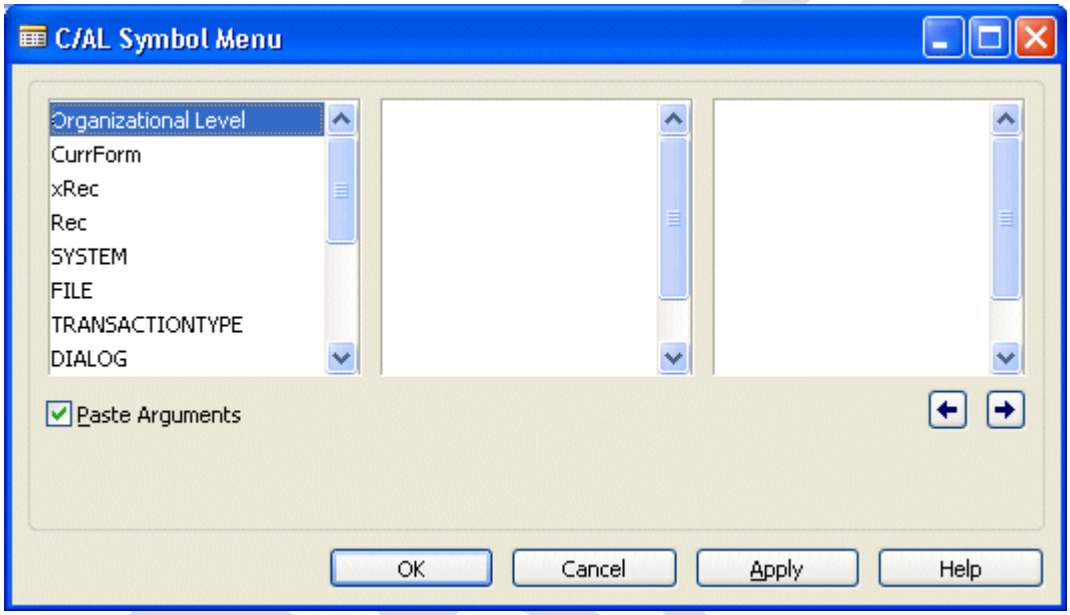
В свойстве SourceExpr нужно указать поле или переменную с типом данных Option, также для данного поля или переменной должно быть заполнено свойство *OptionString*. В данном примере будет использована переменная.

3. Щелкните Вид, C/AL Globals, чтобы открыть окно ***C/AL Globals***.

4. Создайте переменную с именем *Organizational Level* и типом данным Option, затем откройте окно ***Properties*** (SHIFT+F4) для этой переменной.

5. Введите следующий список в поле **Value** свойства *OptionString*: Chief Executive Officer,Chief Financial Officer, Junior Manager,Manager,Salaried Employee,Senior Manager.

6. Откройте окно ***Properties*** (SHIFT+F4) для текстовой зоны и щелкните AssistButton в свойстве *SourceExpr* чтобы открыть окно ***C/AL Symbol Menu***:



7. Выберите переменную Organizational Level и щелкните OK.

8. Сохраните и откомпилируйте форму.

В запущенной форме, список можно раскрыть, щелкнув на кнопку AssistButton в текстовой зоне.

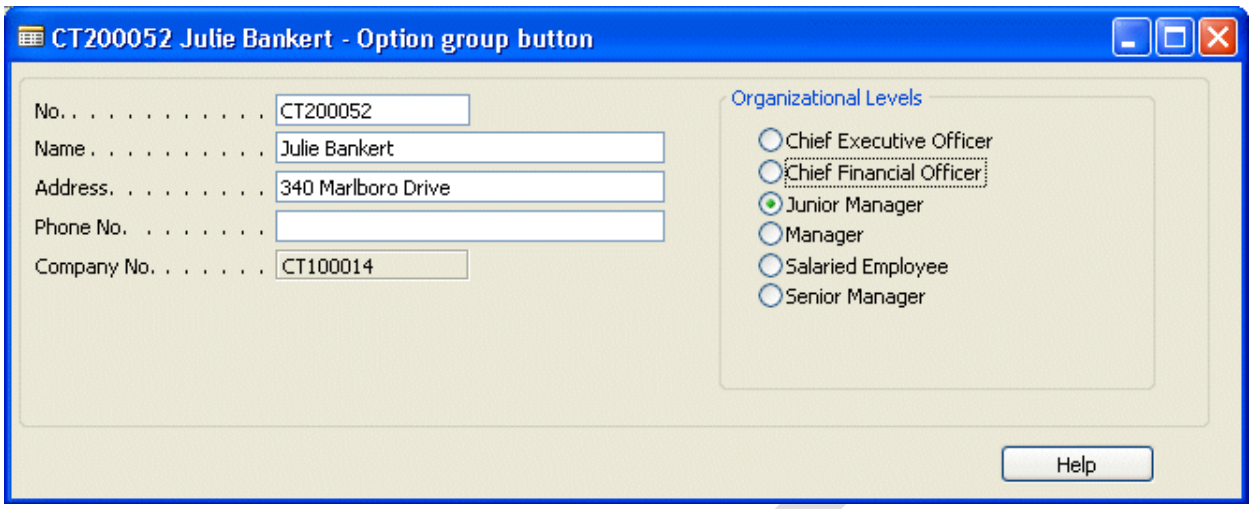
Если текстовая зона с опциями основана на поле таблице, то для вставки текстовой зоны можно воспользоваться окном ***Field Menu***.

> **Примечание**
>
> В текстовую зону можно ввести только те значения, которые указаны в окне ***Properties*** поля или переменной.

В текстовой зоне отображается первая опция. Если в свойстве *OptionString* первая опция пустая, то в текстовой зоне также будет пусто. Это не значит, что можно вводить значения не перечисленные в *OptionString*. В свойстве *OptionString* элемента управления можно указать подмножество опций определенных для поля, добавлять новые опции нельзя.

Создание Групп Кнопок Выбора

Второй способ представления набора опций – это группа кнопок выбора. Функциональность совпадает с той, которую предоставляют выпадающие списки, но визуальное представление отличается. Группа кнопок выбора выглядит следующим образом:



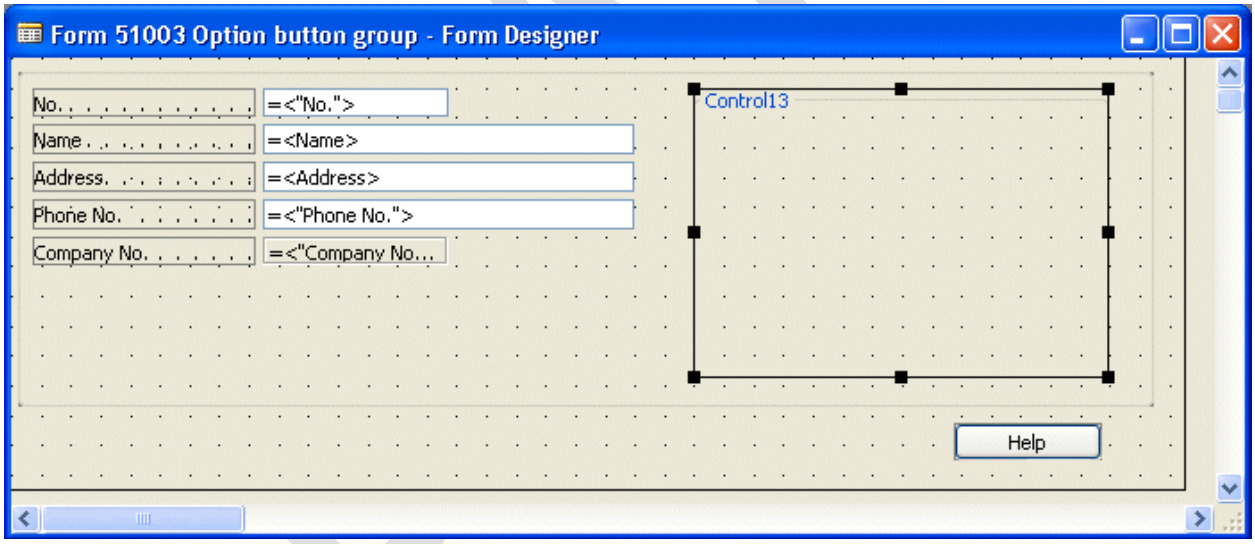
Преимущество использования группы кнопок выбора заключается в том, что пользователь приложения сразу видит все доступные опции и текущую выбранную опцию. Недостатком является то, что расходуется на много больше места, чем при использовании выпадающего списка.

В данном примере будет добавлена группа кнопок выбора на форму, созданную в предыдущем примере.

1. Открыть созданную в предыдущем примере форму в **Form Designer**.

2. Удалить текстовую зону с выпадающим списком.

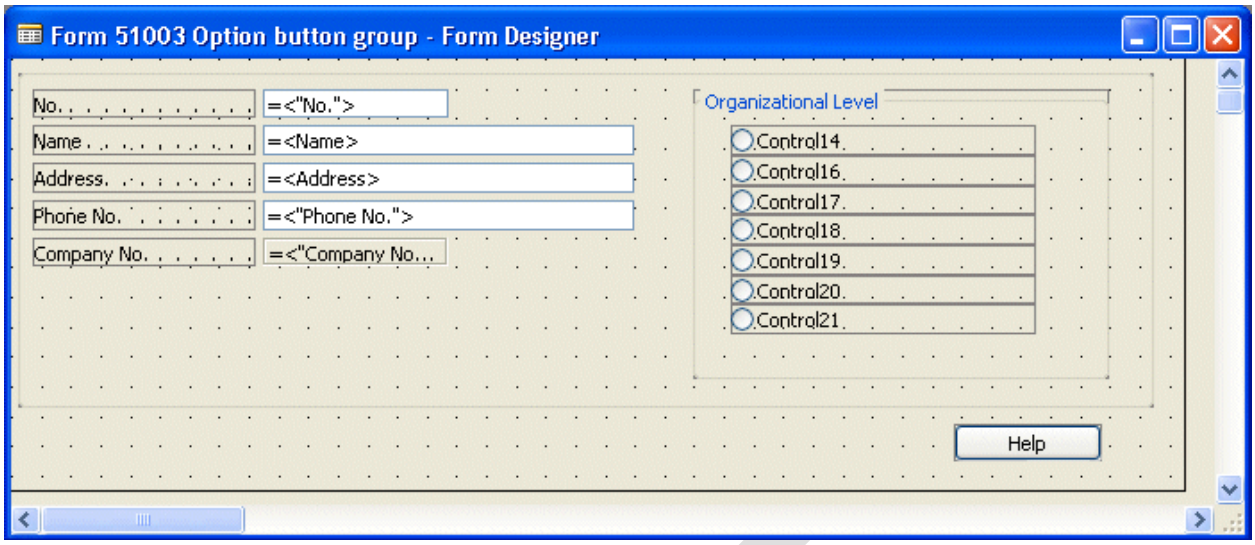
3. Открыть панель инструментов **Toolbox** и добавить на форму фрейм:



4. Открыть окно ***Properties*** и ввести в свойство Caption значение *Organizational Level*.

5. Установить в свойство *BorderStyle* значение BumpUp.

6. Добавить во фрейм шесть кнопок выбора, по одной для каждой опции из переменной *Organizational Level*:



8. В окне ***Properties*** указать в свойстве **SourceExpr** переменную *Organizational Level*.

9. В окне ***Properties*** для каждой опции введите в свойство **OptionValue** соответствующее значение из списка опций переменной *Organizational Level:* Chief Executive Officer,Chief Financial Officer, Junior Manager,Manager,Salaried Employee,Senior Manager. Не забывайте заключать значения в кавычки.

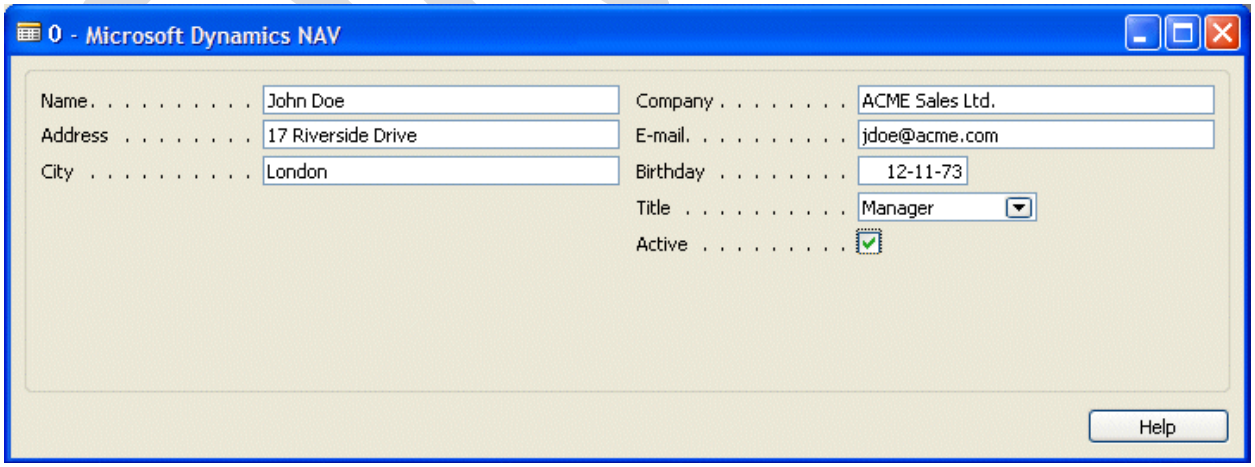
10. Сохранить и откомпилировать форму.

Конечная версия формы должна выглядеть, как показано на рисунке в начале данного раздела.

Так как для всех кнопок выбора указано одинаковое значение свойства **SourceExpr**, то в один момент времени может быть выбрана только одна опция. При выборе какой-либо опции, остальные опции помечаются как не выбранные.

Использование Выключателей (Check Box) для Отображения Значений Типа Booleans

Для отображения типа данных Boolean удобно использовать элемент управления Выключатель. В текстовой зоне Boolean отображается как Да или Нет. В выключателе Да отображается как флажок, а Нет как пустое поле.



Чтобы добавить выключатель:

1. Открыть форму в Form Designer.

2. Если выключатель непосредственно связан с полем таблицы, выберите поле в окне Field Menu. В ином случае создайте несвязанный выключатель.

3. Если вместе с выключателем нужно добавить связанную с ним надпись щелкните инструмент Add Label (по умолчанию для выключателя нет надписи).

4. Выберите инструмент Check Box, затем щелкните по редактируемой форме, чтобы вставить выключатель.

5. Если создан несвязанный выключатель, для связи его с полем или переменной укажите имя переменной в свойстве SourceExpr.

Создание и Использование Командных Кнопок

Командные кнопки полезны для решения ряда задач. Если для создания формы использовался мастер форм, то автоматически на форму добавляется кнопка Help (Справка). Кнопки ДА и НЕТ часто используются в контексте, когда пользователь должен решить должна ли выполняться какая-либо задача или нет. Также командные кнопки используются для запуска других форм или даже других программ.

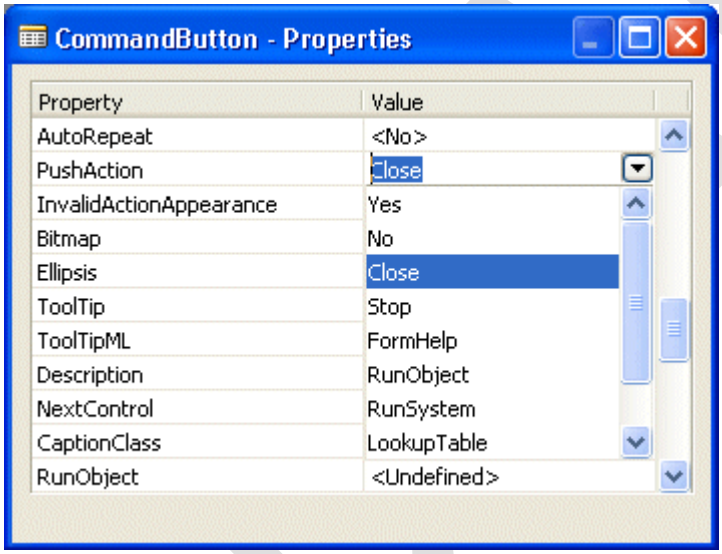
Чтобы добавить командную кнопку:

1. Открыть форму в **Form Designer**.

2. Выбрать инструмент **Command** Button и щелкнуть по редактируемой форме, чтобы вставить командную кнопку. Следующим шагом будет определение действия, которое должно выполняться при нажатии на кнопку.

3. Открыть окно **Properties** для командной кнопки. В свойстве *PushAction* указывается, что должно произойти при нажатии на кнопку.

4. Раскройте выпадающий список возможных значений свойства *PushAction*:



5. Частым назначаемым действием является запуск другой формы. Чтобы при нажатии на командную кнопку запускалась другая форма, выберите значение RunObject из выпадающего списка.

6. В свойстве *RunObject* откройте список объектов и выберите объект, который должен быть запущен при нажатии на командную кнопку.

> **Примечание**
>
> Не для всех значений свойства *PushAction* требуется дополнительная информация. Для значения RunSystem требуется, а для Yes или No - нет.

Хотя данный метод определения действия для командной кнопки довольно прост, он накладывает некоторые ограничения. Например, нельзя передать параметры. Более мощный способ – это использование триггера *OnPush* командной кнопки. Триггеры будут рассмотрены в главе ГЛАВА 11 РАСШИРЕНИЕ ФУНКЦИОНАЛЬНОСТИ ФОРМ.

Добавление Фигур (Shapes) и Изображений

На форму можно добавить графические элементы, такие как фигуры и растровые изображения. Это делается для того, чтобы подчеркнуть информацию, выделив некоторые элементы управления с помощью фигуры, либо же исключительно в декоративных целях.

Использование Фигур (Shapes)

Свойство *ShapeStyle* позволяет выбрать фигуру из набора: прямоугольник, овал, треугольник и пр. Затем для фигуры можно задать толщину и цвет линий с помощью свойств *BorderWidth* и *BorderColor*.

Для создания фигуры:

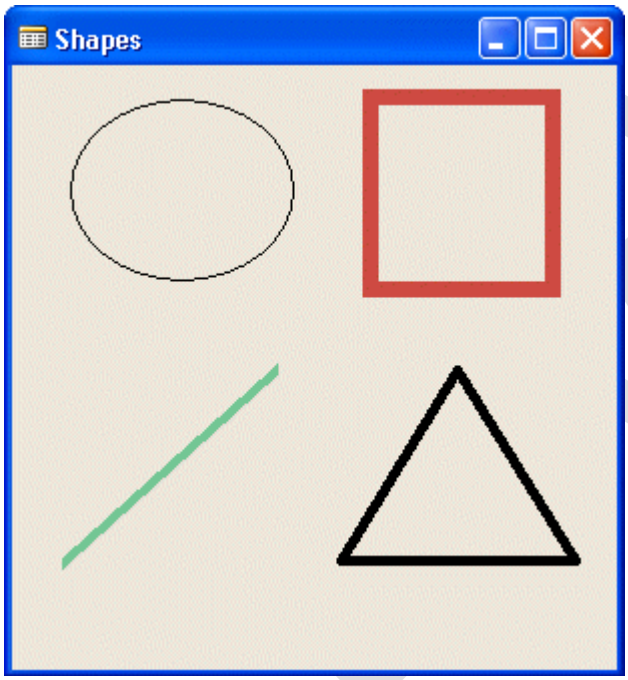
1. Открыть форму в **Form Designer**.

2. Открыть **Toolbox** и выбрать инструмент **Shape**.

3. Щелкнуть и протянуть курсор в редактируемой форме, чтобы задать размер добавляемой фигуры. Размер можно будет скорректировать позже.

4. В окне ***Properties*** фигуры щелкните свойство *ShapeStyle* и выберите подходящую фигуру из выпадающего списка.

5. Используйте свойства *BorderWidth* м *BorderColor* для дальнейшего улучшения вида фигуры, добавленной на форму.



Добавление Статического Изображения (Image)

Простым способом добавления рисунка на форму является использование элемента управления изображение (image).

Чтобы добавить рисунок:

1. Открыть форму в **Form Designer**.

2. Открыть Toolbox и выбрать инструмент **Image**.

3. Щелкнуть и протянуть курсор в редактируемой форме, чтобы создать элемент управления изображение.

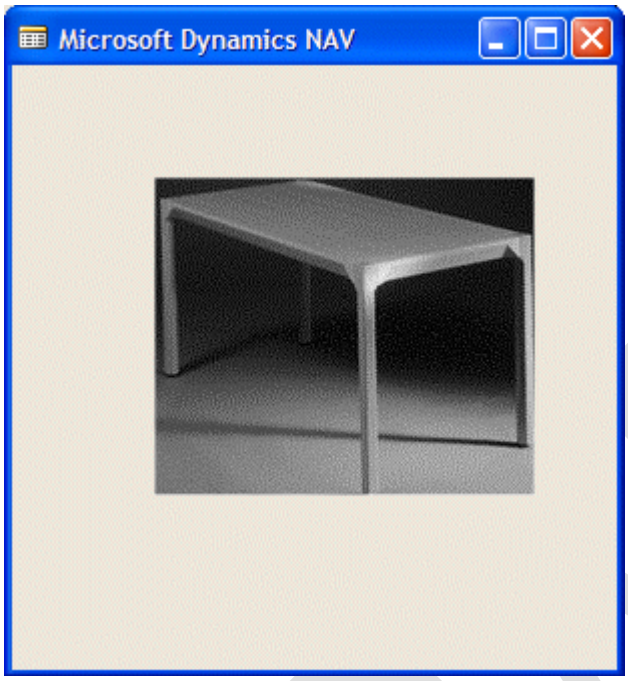
4. Открыть окно Properties для изображения и ввести путь и наименование растрового файла в свойство *Bitmap*.

Растровое изображение будет импортировано, поэтому для приложения не будут нужны внешние файлы. Однако если растровое изображение будет изменено, то нужно будет обновить загруженную копию.

5. Чтобы обновить растровое изображение нужно открыть окно Properties, выбрать свойство *Bitmap* и нажать клавишу F2 при активном поле Value.

Это приведет к тому, что растровое изображение будет загружено повторно.

Размер растрового изображения не может превышать 32 Kb.



Добавление Зон Изображения Зависящих от Данных (Picture Box)

Использование зон изображения (picture box) вместо элемента управления изображение (image) дает дополнительные возможности. В то время как изображение – статический элемент, зону изображения можно изменять динамически. Например, можно создать список растровых изображений и выбирать из него изображения во время работы (общий объем всех изображений в списке может быть 32 Kb).

Другой возможностью являет отображения с помощью зоны изображения рисунков, хранящихся в BLOB полях. Поля BLOB могут хранить до 2 GB.

Чтобы добавить зону изображений:

1. Открыть форму в **Form Designer**.

2. Открыть **Toolbox** и выбрать инструмент **Picture Box**.

3. Щелкнуть и протянуть курсор в редактируемой форме, чтобы создать элемент управления зона изображения.

4. Открыть окно **Properties** для зоны изображения.

Чтобы создать список изображений, введите пути и имена растровых изображений, разделенные точкой с запятой. Размер всех изображений списка не должен превышать 32 Kb.

Система предоставляет серию стандартных изображений, к которым можно обращаться по номерам от 1 до 53

5. Значение свойства *SourceExpr* указывает, какое растровое изображение выбрано: для первого изображения в списке используется значение 1, и т.д. Если в *SourceExpr* рассчитано значение за

DRAFT

пределами диапазона изображений, никакое изображения отображаться не будет. Позже будет объяснено, как вводить выражения на языке C/AL в свойство *SourceExpr*.

Чтобы отображать изображения, хранящиеся в BLOB полях, наименование BLOB поля нужно ввести в свойство *SourceExpr*. Не заполняйте свойство *BitmapList*.

Рисунки на Командных Кнопках, Кнопках Меню, Кнопках Выбора и Выключателях

На командных кнопках, кнопках меню, кнопках выбора и выключателях можно отображать растровые изображения либо вместо заголовка, либо вместе с ним.

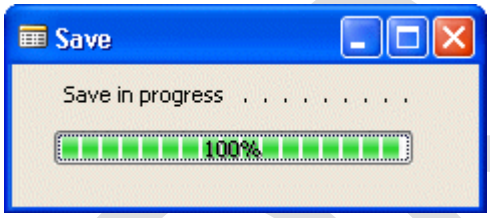
У всех указанных элементов управления есть свойство *Bitmap*. В нем можно указать путь и название растрового изображения. Максимальный размер растрового изображения - 32 Kb, и оно импортируется в систему, а не ссылается на внешний файл. Это означает, что если исходное растровое изображение было изменено, то его нужно загрузить повторно.

Также у элементов управления есть свойство *BitmapPos*, где указывается выравнивание растрового изображения в элементе управления. Это полезно при совместном использовании заголовка и растрового изображения.

Использование растровых изображений на кнопках или выключателях делает интерфейс интуитивно понятным. Хорошо-подобранное изображение воспринимается и запоминается лучше, чем надпись.

Использование Индикаторов для Отображения Значений

Элемент управления индикатор (indicator) позволяет графически представить данные. Чтобы система могла рассчитать шкалу нужно ввести минимальное и максимальное значение. Если эти значения не указаны, система использует значения по умолчанию (см. онлайновую справку *C/SIDE Reference Guide*).



Чтобы создать индикатор:

1. Открыть форму в **Form Designer**.

2. Выбрать инструмент **indicator**.

3. Щелкнуть и протянуть курсор в редактируемой форме, чтобы создать элемент управления индикатор.

4. Укажите в свойстве *SourceExpr* значение, которое будет управлять индикатором.

5. Установите свойства *MinValue* и *MaxValue*.

6. Установите значение свойства *Percentage,* чтобы указать должен ли появляться символ «%». Независимо от выбранного значения шкала остается прежней. Процент рассчитывается по следующей формуле:

((value of SourceExpr) – MinValue) / (MaxValue – MinValue)*100.

DRAFT

ГЛАВА 11 РАСШИРЕНИЕ ФУНКЦИОНАЛЬНОСТИ ФОРМ

В данной главе раскрывается далее тема создания и редактирования форм.

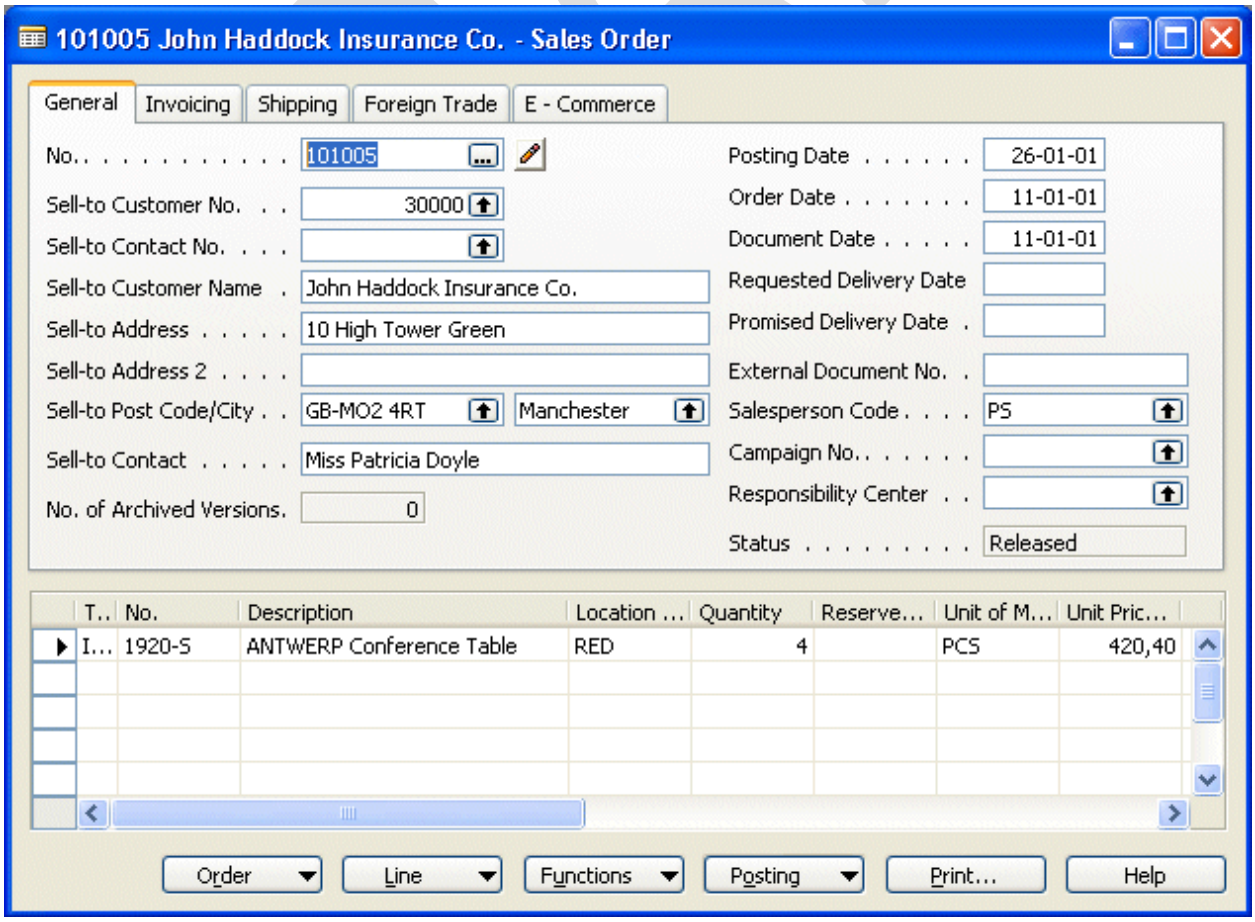
- Главные и сопутствующие формы
- Выбор значений и проверка значений
- Детализация значений
- Вызов другой формы
- Создание кнопок меню
- Триггеры форм и элементов управления
- Типы форм и их характеристики

11.1 Главные и Сопутствующие Формы

Правильно спроектированные базы данных не содержат избыточную информацию, вместо этого используются связи между таблицами. Наиболее часто встречается связь «Один-ко-Многим».

Например, ваше приложение обрабатывает заказы продажи, в одном заказе продажи может быть много товаров. Некоторая информация заказа продажи, такая как адрес клиента, относится к заказу, другая, например, номера товаров – к строкам. В правильно спроектированной базе данных, где нет избыточной информации, это означает, что информация хранится в двух таблицах: одна таблица-заголовок, общей информацией по заказу, вторая таблица – строки, в ней содержится информация по каждому отгружаемому товару. Между таблицами установлена связь «Один-ко-Многим».

Однако пользователям приложения нужно одновременно видеть информацию из обеих таблиц. Информация из заголовка, вместе с информацией из строк может выглядеть так:



Хотя это похоже на обычную форму, на самом деле здесь две формы.

Главная форма (main form) это «Один» из связи «Один-ко-Многим». В данном примере, она основана на таблице ***Sales Order Header***. Сопутствующая форма (subform) это «многие» из связи «Один-ко-Многим», она основана на таблице ***Sales Order Line***. При выборе заказа продажи в главной форме, содержимое сопутствующей формы изменяется, чтобы отображать только те строки продажи, которые связаны с выбранным заголовком продажи. Чтобы информация синхронизировалась, между главной формой и сопутствующей формой есть связь.

Создание Главной Формы

Для создания главной формы не применяются никакие специальные методы. Чтобы установить связь главной формы с сопутствующей, нужно добавить элемент управления Сопутствующая форма (subform). Хотя с помощью элемента управления Сопутствующая форма устанавливается связь, он не является формой. Если в качестве сопутствующей формы требуется использовать существующую форму, выполните шаги, описанные в данном разделе. Если в качестве сопутствующей формы должна использоваться новая форма, удобнее начать именно с создания сопутствующей формы. Как это сделать будет написано в следующем разделе.

Чтобы создать главную форму из связи главная форма/сопутствующая форма нужно:

1. Открыть существующую форму в **Form Designer**.

2. Открыть **Toolbox** и выбрать инструмент **Subform**.

3. Щелкнуть и протянуть курсор в редактируемой форме, чтобы создать элемент управления сопутствующая форма.

4. Открыть окно **Properties** (SHIFT+F4) для сопутствующей формы.

5. В свойстве *SubFormID* указать наименование формы, которая будет использоваться как сопутствующая (также можно выбрать из списка)

6. В свойстве *SubFormLink* ввести выражение, которое будет связывать две таблицы (например, общие для двух таблиц поля). Для ввода выражения можно использовать функцию assist-edit (щелкните кнопку AssistButton). В поле Field укажите названия полей со стороны «много» связи «Один-ко-Многим» (поля таблицы-источника для сопутствующей формы). Затем укажите значение FIELD в поле Type. В поле Value укажите поля со стороны «один» связи «Один-ко-Многим» (поля таблицы-источника для главной формы).

7. В свойстве *SubFormView* можно указать ключ, порядок сортировки и фильтр таблицы, которые будут применяться к таблице, когда она отображается в сопутствующей форме (не обязательно указывать что-либо в данном свойстве)

> **Пример**
>
> В свойстве *SubFormLink* кроме типа FIELD можно выбирать и другие типы записей. Если указан тип CONST, поле Value должно содержать выражение-константу, в этом случае будут отобраны записи соответствующие данному выражению. Если выбран тип FILTER, то в поле Value нужно указать выражение-фильтр, например, 10|30..40.

Создание Сопутствующей Формы

Для сопутствующих форм не предъявляются особые требования, однако надо помнить, что создаваемая форма будет использоваться для отображения «многих» из связи «один-ко-многим», а для этого подходят не все формы.

Обычно сопутствующая форма – это табличная форма, с табличной зоной. Табличная зона должна заполнять форму полностью, поэтому и для табличной зоны сопутствующей формы и для элемента управления сопутствующая форма главной формы нужно установить в свойства *HorzGlue* и *VertGlue* значение Both. Это гарантирует, что сопутствующая форма и табличная зона будут изменять размеры при изменении размеров главной формы.

DRAFT

Советы

Ниже приведено несколько советов по созданию сопутствующих форм:

- В первую очередь следует создать сопутствующую форму. Затем переписать значения высоты и ширины сопутствующей формы (см. свойства width и height в окне ***Properties)*** и установить такие же значения для элемента управления сопутствующая форма в главной форме.
- Обычно если сопутствующая форма является табличной формой, она будет выглядеть лучше, если табличная зона полностью заполняет форму по вертикали. Это гарантирует, что вокруг таблицы не будет лишнего свободного места. Установите в свойства *HorzGlue* и *VertGlue* значение Both.
- Если сопутствующая форма является табличной формой, следует установить размер формы так, чтобы одновременно на ней отображалось не большое количество полей. Для элемента управления сопутствующая форма в главной форме нужно установить в свойства *VertGlue* и *HorzGlue* значение Both. Теперь можно будет изменять главную форму по вертикали и по горизонтали, при этом будет изменяться размер сопутствующей формы, и она будет отображать то больше, то меньше полей, в зависимости от размера.

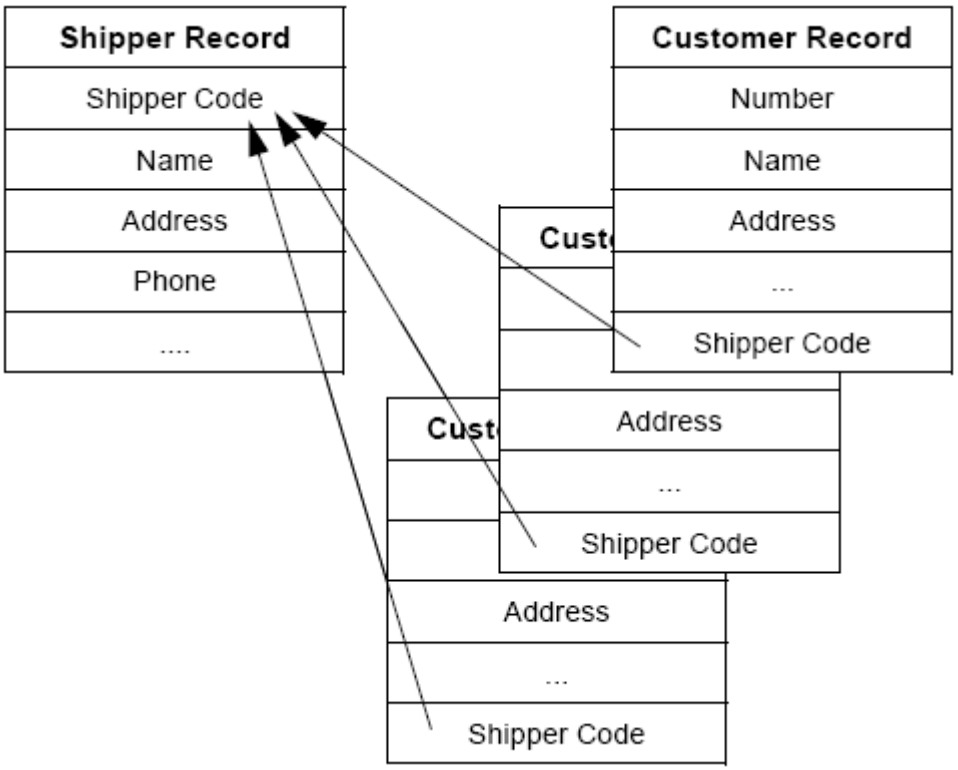
11.2 Выбор Значений и Проверка Значений

В предыдущем разделе было описано, как создавать главную и сопутствующую форму для отображения данных из связи «Один-ко-Многим». Главная форма представляла часть «один», а сопутствующая форма – «многие».

Теперь предположим, что разрабатывается форма, которая связана с таблицей, содержащей информацию о клиенте. Некоторая информация является уникальной для каждого покупателя, например, наименование и адрес. Кроме того, требуется хранить информацию об агенте по доставке, с помощью которого обычно доставляется товар для каждого покупателя. Всего существует несколько агентов по доставке, поэтому хранение информации, связанной с агентом, в записи клиента будет являться нарушением правил проектирования реляционных баз данных.

Вместо этого должна быть создана отдельная таблица ***Shipper***, а в таблицу ***Customer*** должно быть добавлено поле **Shipper Code.** Данное поле будет использоваться для связи каждой записи и таблицы ***Customer*** с таблицей хранящей всю подробную информацию об агенте по доставке(***Shipper)***.

DRAFT



Каждый клиент может быть связан только с одним агентом, в то же время агент по доставке может быть связан с несколькими различными клиентами. В данной ситуации использование структуры Главная Форма/Сопутствующая Форма не применимо (хотя она может использоваться в карточке агента, в сопутствующей форме могут отображаться все клиенты, связанные с выбранным агентом по доставке).

При создании формы ***Customer*** (и таблицы) нужно ответит на два вопроса:

- Должна ли система предоставлять простой способ вводить код агента по доставке?
- Должна ли система проверять, чтобы значение, введенное в поле **Shipper Code** таблицы ***Customer,*** существовала в таблице ***Shipper***? (В терминах системы такая проверка называется Validate).

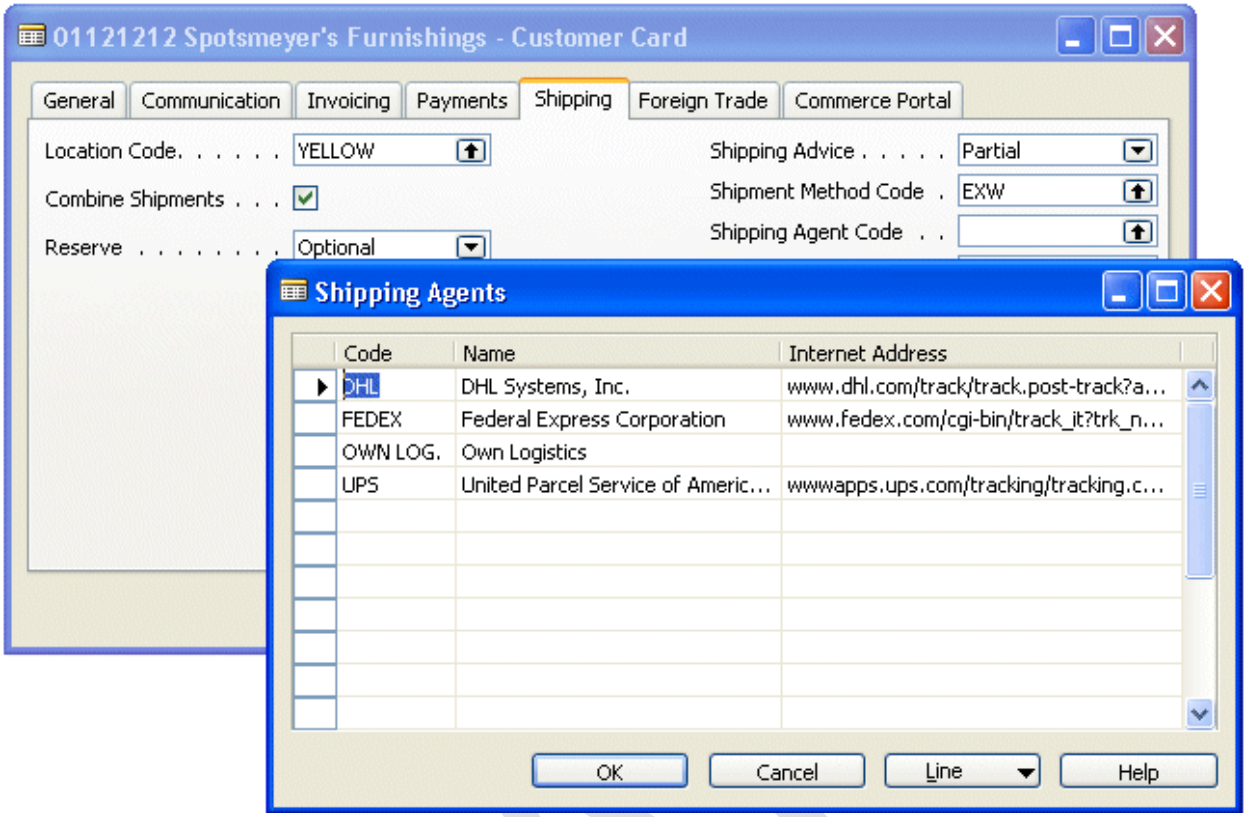
Если не установить связь с таблицей ***Shipper***, то при заполнении поля **Shipper Code** нужно помнить коды всех агентов. Это значит, что легко можно ошибиться и ввести код, который не существует в таблице ***Shipper***. Если таблицы связаны, система предоставляет возможность выбора значения из таблицы ***Shipper***. Для этого нужно нажать клавишу F6 или щелкнуть кнопку Выбор, затем выбрать код агента из появившегося списка, где кроме кодов будут указаны наименования и адреса.

Элементы управления могут быть связаны с полями другой таблицы двумя различными способами. Первый способ, это создание связи на уровне таблицы (в свойствах поля **Shipper Code** таблицы ***Customer***). Второй способ, это создание связи на уровне формы (в свойствах текстовой зоны, которая отображает код агента на форме ***Customer***).

Если нужно удостовериться, что введенные в таблицу ***Customer,*** коды агентов по доставке существуют, можно настроить систему так, чтобы она выполняла соответствующую проверку. Для этого используется свойство *ValidateTableRelation,* оно доступно как для поля таблицы, так и для текстовой зоны формы и позволяет указать, должно ли существовать введенное значение в таблице ***Shipper***.

Кроме простой проверки того, что введенный код существует в таблице ***Shipper*** можно задать более сложные правила проверки значения на основании комбинации полей в обеих таблицах. Например, система может выполнять проверку, работает ли данный агент по доставке в стране/регионе клиента. Для этого нужно создать правила проверки на языке C/AL и поместить их в триггер *OnValidate* элемента управления формы.

Форма клиента с активной функцией выбора значения для поля **Shipping Agent Code** из таблицы ***Shipper*** выглядит следующим образом:



Создание Связей между Таблицами

Связи между таблицами могут быть определены в двух местах: в Описании Таблицы или в Описании Формы. В обоих случаях, связь задается в свойстве *TableRelation* поля или элемента управления. Функционально нет разницы между связью, заданной на уровне таблицы, и связью, заданной на уровне формы. Однако при проектировании приложения разница есть. Если связь установлена на уровне таблицы, у всех текстовых зон, которые связаны с данным полем будет доступна функциональность выбора, при этом не нужно будет прилагать дополнительные усилия. Данную функциональность можно запретить, для этого установите в свойстве текстовой зоны *Lookup,* значение Нет.

Чтобы установить связь между таблицами:

1. Открыть форму в Form Designer.

2. Открыть окно Properties (SHIFT+F4) поля или элемента управления.

3. Щелкнуть кнопку AssistButton в поле **Value** свойства *TableRelation*. C/SIDE отобразит окно ***Table Relation***.

4. В поле **Table** введите название таблицы, с которой должна быть установлена связь, ее можно выбрать из списка.

5. В поле **Field** указывается наименование поля, его также можно выбрать из списка полей таблицы, указанной в поле **Table**.

Поля **Condition** и **Table Filter** используются для создания сложных связей. Например, в зависимости от значения в поле таблицы функционал Выбор будет открывать различные таблицы. Каждая строка условия, аналогичная выражению *if then...else if*

При необходимости, можно указать фильтр таблицы (для таблицы указанной в поле **Table**), для этого используется поле **Table Filter**.

Проверка Значений

Если нужно удостовериться, что введенные в таблицу значения существуют в связанной таблице, то это легко сделать, установив в свойство *ValidateTableRelation,* значение Да. Данное свойство доступно как для поля таблицы, так и для текстовой зоны формы.

Если требуется применять более сложные правила проверки значения, то для этого можно создать правила проверки на языке C/AL и поместить их в триггер *OnValidate* поля или элемента управления формы.

Использовать Функцию Выбор по Умолчанию или Создать Свою Собственную?

Если при использовании функции Выбор нужны дополнительные возможности по управлению, например использование условий и фильтров, то можно написать свою функцию на языке C/AL в триггере *OnLookup.* Это позволит полностью переопределить функцию Выбор.

Какая функция Выбор будет использоваться, определяется следующими правилами:

- Триггер на уровне формы имеет преимущество перед триггером на уровне таблицы.
- Оба триггера имеют преимущество перед системной функцией Выбор, используемой по умолчанию.

Определение Формы для Функции Выбор

При использовании системной функции Выбор, нужно определить какая форма при этом должна отображаться. Определить форму можно двумя способами:

- Для каждой таблицы можно указать форму, которую следует использовать по умолчанию для функции Выбор. Эта форма указывается в свойстве *LookUpFormID* таблицы.
- Форму можно указать в свойстве *LookUpFormID* элемента управления, для которого будет использоваться функция Выбор.

Если заданы оба свойства, то будет использоваться форма, определенная в свойстве элемента управления.

Если форма не задана (ни для таблицы, ни для элемента управления) то, несмотря на наличие кнопки Выбор в текстовой зоне, при щелчке по кнопке не будет выполнено никаких действий. Если вы написали собственную функцию в триггере *OnLookup,* то форму надо явно указать в функции RUNMODAL.

> **Совет**
>
> Если для таблицы всегда создавать табличную форму и указывать эту форму в свойстве LookupForm (и DrillDownForm) таблицы, вы никогда не забудете создать форму выбора. Если позже будет решено, что данная форма не подходит для некоторых задач, в свойствах элемента управления можно будет указать другую форму.

Permanent Assist

Это свойство элемента управления. Если в нем установлено значение Да, кнопка Выбор отображается постоянно, в противном случае кнопка Выбор отображается, только когда на элементе управления установлен фокус.

Выбор в Текущей Таблице

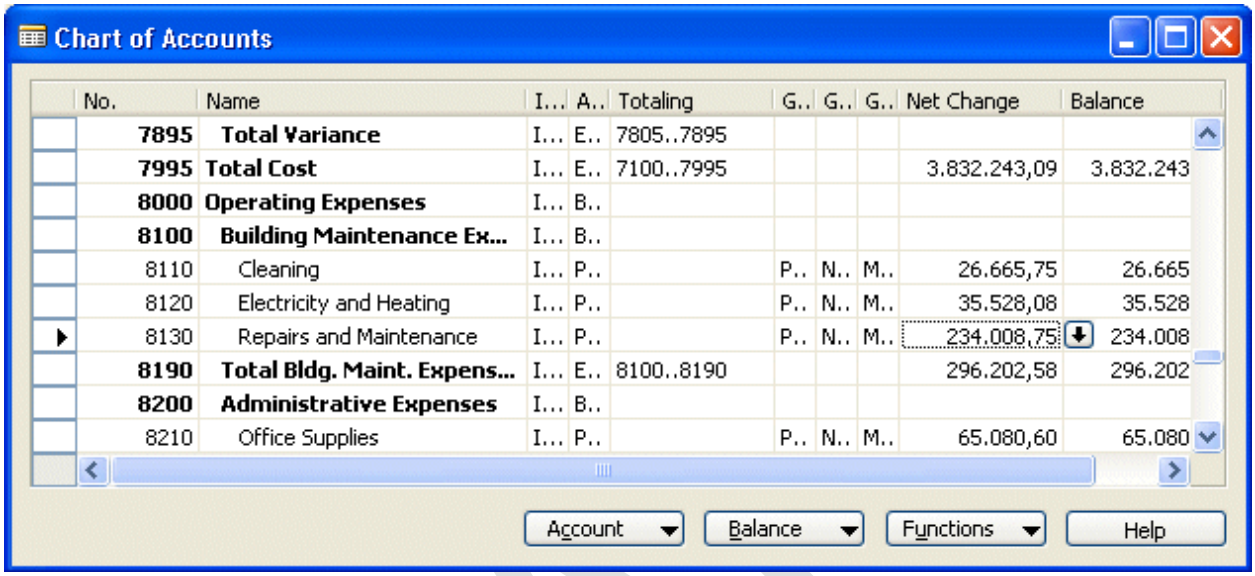
Для выбора значения из текущей таблицы следует использовать действие LookupTable (применяется к командным кнопкам и элементам меню). При щелчке по такой командной кнопке (элементу меню), система отобразит список всех записей текущей таблицы, из которой можно выбрать нужную запись. Эта запись станет текущей записью формы.

DRAFT

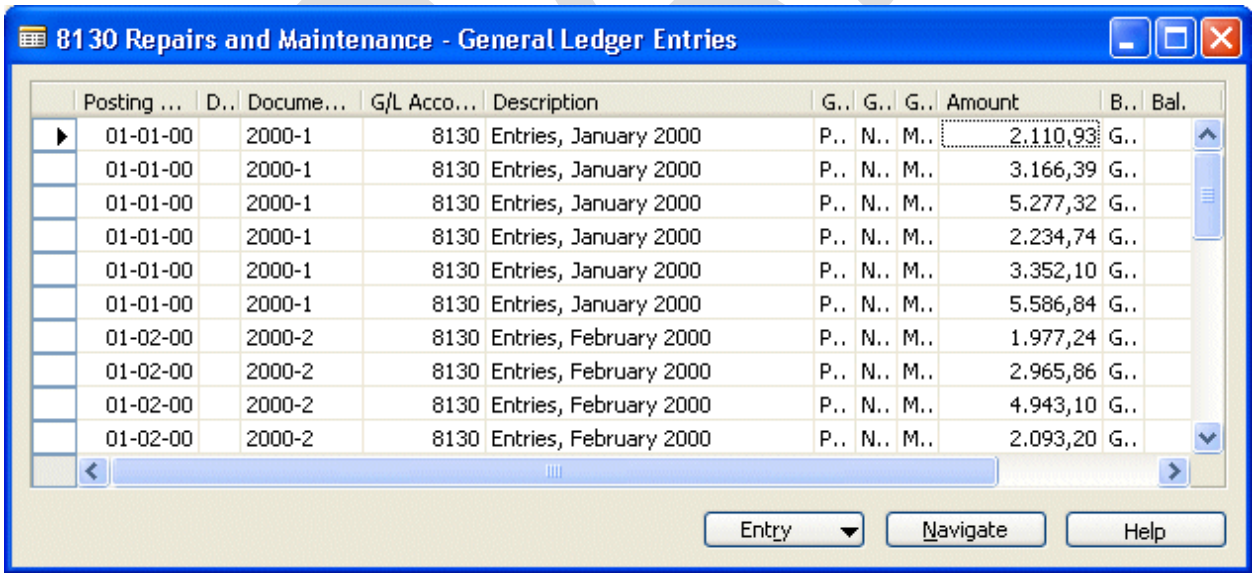
11.3 Детализация Значений

В главе «Таблицы – Базовые Понятия» было рассмотрено понятие вычисляемых полей (FlowField). Если текстовая зона связана с вычисляемым полем, у нее автоматически появляется кнопка Детализировать (*drill-down).* Если щелкнуть по данной кнопке система отобразит все записи, на основании которых рассчитано значение, отображаемое в вычисляемом поле.

В окне **Chart of Accounts (План Счетов)** можно щелкнуть кнопку Детализировать в поле **Net Change.** Это вычисляемое поле, суммирующее все транзакции по данному счету:



Откроется окно ***General Ledger Entries*** (Фин. Книга Операций), в котором будет показан список всех транзакций, на основании которых рассчитано значение в поле **Net Change**:



Возможности по детализации значений предоставляются текстовым зонам, связанным с вычисляемыми полями. Для этого не требуется выполнять какие-либо действия при создании формы, кроме определения свойства DrillDownFormID (его можно задать как на уровне таблицы, так и на уровне элемента управления формы).

Функция Детализировать во многом похожа на функцию Выбор, и с ней можно выполнять большинство действий используемых при работе с функцией Выбор. Единственным исключением является то, что функция Детализировать применятся только для вычисляемых полей, которые определяются на этапе проектирования таблиц.

Настройка Функции Детализировать (Drill-down)

Функцию Детализировать можно настроить следующими способами:

- Полностью отключить функцию Детализировать. Для этого в свойстве *DrillDown* текстовой зоны установить значение Нет.
- Текстовые зоны, базирующиеся на одном и том же вычисляемом поле, могут открывать разные формы для детализации, если в свойстве DrillDownFormID текстовых зон указать различные формы.
- Можно указать будет ли кнопка Детализировать видна постоянно или только когда на элемент управления установлен фокус. Для этого нужно использовать свойство текстовой зоны *PermanentAssist.* Если в нем установлено значение Да, кнопка Выбор отображается постоянно, в противном случае кнопка Выбор отображается, только когда на элемент управления установлен фокус
- Можно изменить поведение функции Детализировать, написав свою функцию на языке C/AL в триггере *OnDrillDown* поля таблицы или элемента управления формы. В этом случае запускать форму нужно из C/AL кода триггера.

11.4 Запуск Других Форм

Функция Выбор используется для выбора значения из другой таблицы, обычно при этом отображается только часть полей связанной таблицы.

В некоторых случаях было бы удобно обновить некоторые поля связанной таблицы, не закрывая форму, из которой была запущена функция Выбор.

Например, при приеме заказов по телефону, использование функции Выбор удобно при выборе номера клиента для заказа продажи. Однако клиент может позвонить, чтобы заказать что-нибудь, и сообщить при этом, что они изменили адрес. Чтобы внести измененную информацию потребуется закрыть заказ продажи, открыть карточку клиента, найти нужного клиента, изменить адрес и снова открыть заказ продажи. Это может занять некоторое время.

Удобнее было бы открывать карточку клиента прямо из заказа продажи, чтобы при этом автоматически открывалась соответствующая запись в таблице клиентов. При этом можно будет изменить адрес, не закрывая заказа продажи.

Чтобы запустить другую форму нужно добавить элементу управления, для которого доступно свойство *PushAction.*Таким образом, при нажатии на элемент управления можно будет запускать карточку клиента с параметрами, позволяющими сразу открыть правильную запись. Для этих целей можно использовать командные кнопки, элементы меню, выключатели и кнопки выбора.

Добавление Командной Кнопки

Чтобы добавить командную кнопку для запуска формы следует:

1. Добавить командную кнопку (процедура была описана в разделе 10.8 Как Использовать Элементы Управления в Приложениях.

2. Установить в свойстве *PushAction* командной кнопки значение RunObject.

3. Установить в свойстве *RunObject* название вызываемой формы. Так как с помощью RunObject можно запустить любой объект, нужно также указать тип запускаемого объекта (Form, Codeunit и т.д.). Объект можно выбрать из списка, в этом случае тип объекта будет вставлен автоматически.

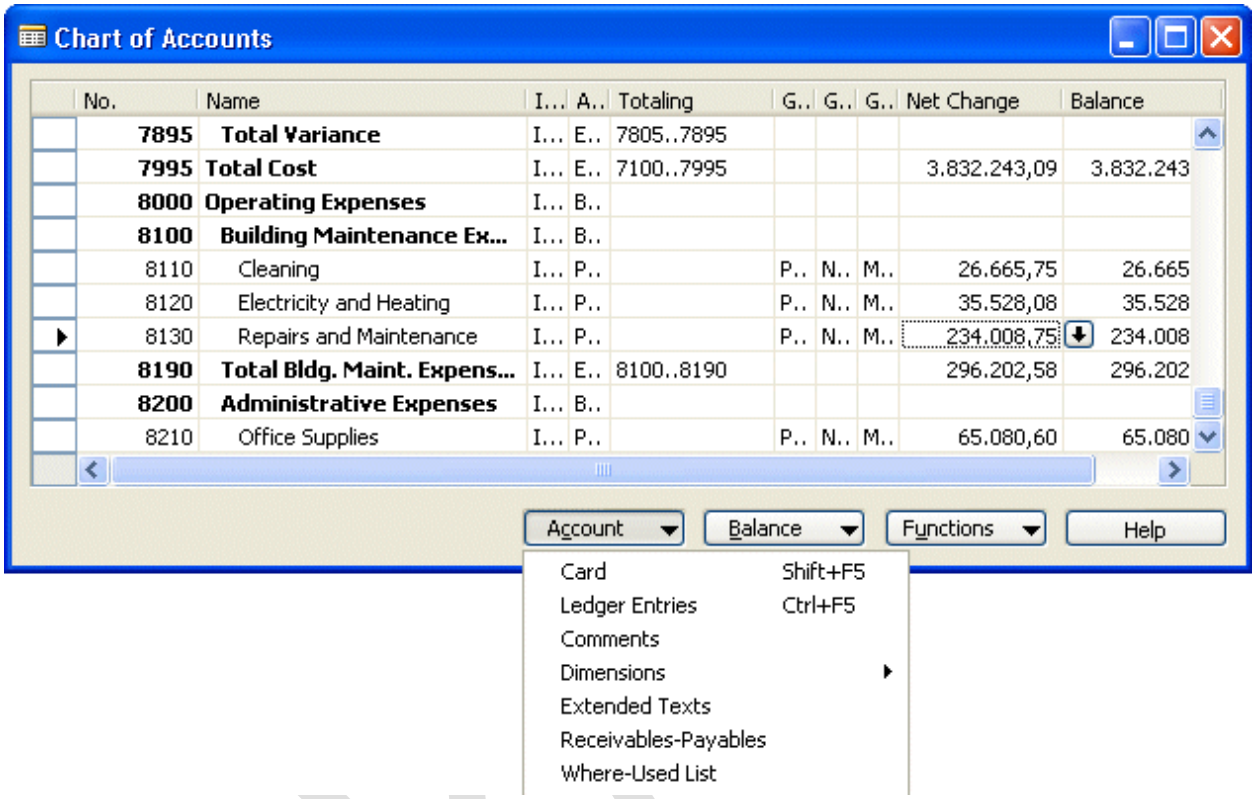
4. В свойстве *RunFormLink* указывается связь с запускаемой формой. Для создания выражения можно использовать кнопку AssistButton. В поле **Field** указывается поле таблицы, на которой основана запускаемая форма. В поле **Type** выберите тип связи FIELD. В поле **Value** укажите поле в таблице-источнике для текущей формы, данное поле должно соответствовать полю из второй таблицы.

DRAFT

11.5 Создание Кнопок Меню

Хотя командные кнопки предоставляют удобный способ расширения функциональности формы, слишком большое количество кнопок на форме делает ее загроможденной и затрудняет ее использование. Если на форме много командных кнопок, следует задуматься над замещением их несколькими кнопками меню.

При нажатии на кнопку меню, появляется меню:



Каждая строка меню называется Элемент меню (*menu item)*. Элементы меню могут:

- Выполнять действия при нажатии на элемент меню. Это могут быть действия аналогичные действиям командных кнопок или действия, написанные на языке C/AL. У элементов меню, также как у командных кнопок есть триггер *OnPush*.
- Содержать подменю.
- Использоваться в качестве разделителя (линии, которая объединяет вместе несколько элементов меню).

Добавление Кнопок Меню на Форму

Созданием меню включает два шага. Первый – это добавление на форму кнопки меню, процедура аналогична добавлению командной кнопки. Затем нужно открыть Menu Designer для кнопки меню и создать элементы меню.

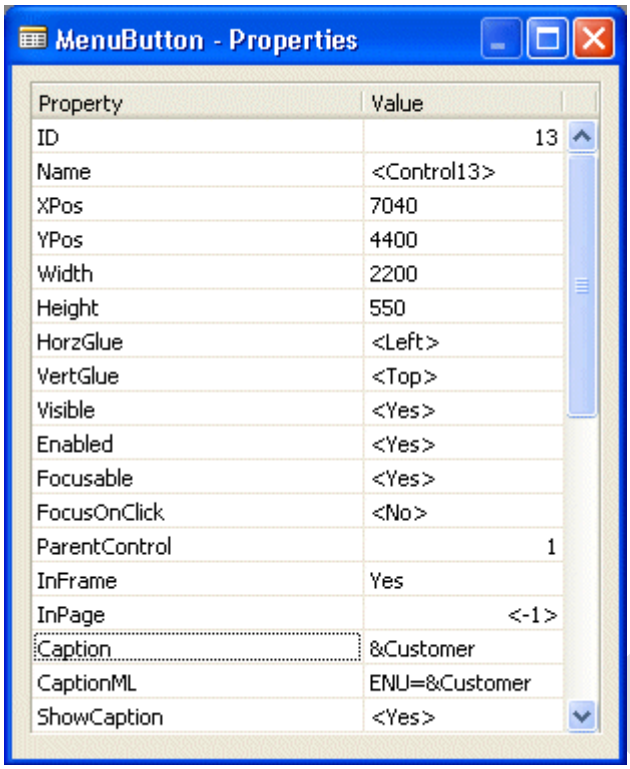
Чтобы добавить кнопку меню надо:

1. Открыть форму в **Form Designer**.

2. Открыть панель инструментов **Toolbox** и выбрать инструмент **Menu Button**.

3. Вставить кнопку меню на форму.

4. Выбрать кнопку меню и открыть окно **Properties** (SHIFT+F4). Так как кнопка меню не связана с данными (полем или переменной) в свойствах *Name* и *Caption* указаны значения по умолчанию (например, *Control7*). Измените заголовок (caption) на подходящий текст. Если текст содержит символ (&), система будет считать следующую букву ускоряющей клавишей.

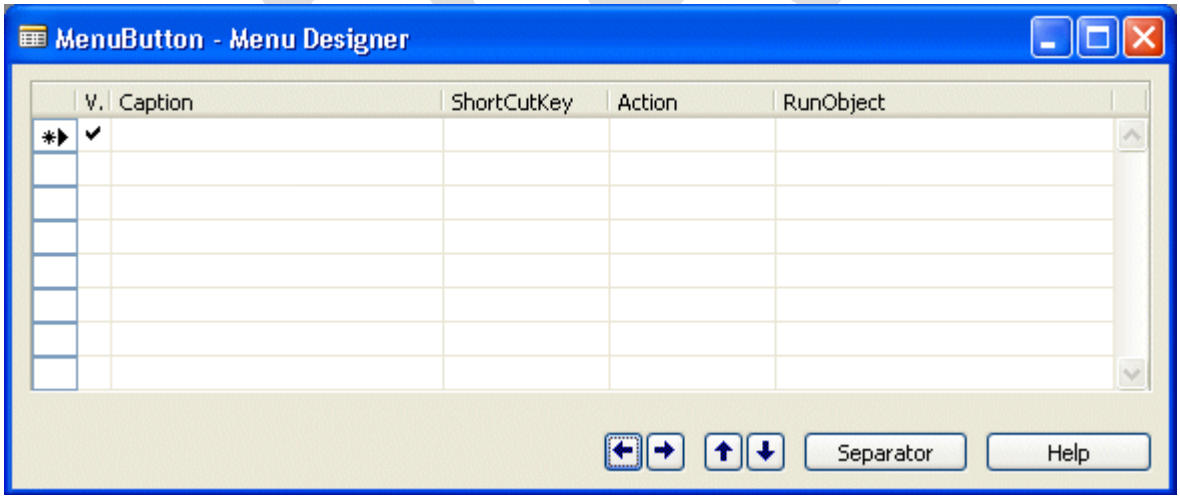


Добавление Элементов Меню

После того как кнопка меню была создана и изменена согласно процедуре приведенной выше, к ней можно добавить элементы меню.

Чтобы добавить элементы меню:

1. Выбрать кнопку меню и щелкнуть Вид, Menu Items, чтобы открыть ***Menu Designer***.



2. Первое поле **Visible** по умолчанию содержит значение Да. Оставьте его без изменений.

3. Заполните поле **Caption,** чтобы добавить элемент меню. Если не определять клавишу быстрого выбора (используя символ & в свойстве Caption), система автоматически будет использовать в качестве клавиши быстрого доступа первую букву в поле Caption. Если в меню несколько элементов начинаются на одну букву, следует задать клавишу быстрого выбора вручную.

4. Также клавишу быстрого выбора можно задать в поле **ShortCutKey**. Для этого есть три:

Клавиша	Вводить как
Функциональные клавиши	F1, F2, F3,...

Клавиша	Вводить как
CONTROL, ALT, SHIFT	CTRL, ALT, SHIFT
Другие клавиши	A, B, C,… (должны быть частью комбинации клавиш с CTRL или ALT).
Комбинации клавиш	Например: CTRL+A, SHIFT+F2

Клавиши быстрого выбора активны, пока фокус установлен на форму, к которой принадлежит кнопка меню. Старайтесь не использовать одинаковые комбинации клавиш для выполнения различных действий на разных формах, так как это приносит путаницу. Также не стоит использовать комбинации клавиш, которые уже используются системой.

5. В поле **Action** указывается действие, которое будет выполняться при нажатии на элемент меню. Действие можно выбрать из выпадающего списка. Как вариант можно написать код на языке C/AL в триггере *OnPush*.

Если выбрано действие RunObject, можно указать запускаемый объект, это может быть форма, отчет или программный модуль.

6. Щелкните кнопку выбор в поле **RunObject,** чтобы выбрать запускаемый объект. Чтобы настроить параметры для других действий (например, RunSystem) нужно использовать окно ***Properties*** элемента меню. Об этом будет рассказано в следующем разделе.

Добавление Других Элементов Меню

Кроме элементов меню, при нажатии на которые выполняются определенные действия, меню могут включать разделители и подменю.

Разделители (Separators)

Разделитель – это горизонтальная линия в меню, которую нельзя выбрать, и которая не выполняет никаких действий. Разделители помогают сгруппировать элементы меню. Чтобы добавить разделитель щелкните по строке, после которой он должен быть установлен, затем щелкните кнопку **Separator** в **Menu Designer**.

Подменю и уровни меню

Система позволяет использовать вложенные меню, т.е. при выборе одного элемента меню открывается другое меню.

В **Menu Designer** подменю создаются с помощью отступов (кнопка со стрелкой вправо). Элементы меню, для которых задан отступ, становятся элементами меню в подменю.

В окне ***Properties*** элемента меню можно увидеть, что при создании элемента меню в свойстве *MenuLevel* установлено значение 0 (ноль). После того как для элемента меню был добавлен отступ, в свойстве MenuLevel устанавливаются значения 1, 2, 3 и т.д. (по единице на каждый уровень). Чтобы уменьшить вложенность нужно щелкнуть по кнопке со стрелкой влево, каждое нажатие уменьшает уровень на 1.

При создании подменю нужно следовать следующим логическим правилам:

- Как минимум у одного элемента меню в свойстве MenuLevel должно стоять значение 0 (ноль).
- Значение MenuLevel может быть больше предыдущего только на один уровень.
- Большее значение свойства MenuLevel идет за меньшим значением (например, 1 идет после 0). Элементы меню с MenuLevel = 0 открывает подменю, а элемент с MenuLevel = 1 становится элементом меню в данном субменю. Для элемента меню, который открывает подменю, нельзя задать никаких действий.
- Система поддерживает 10 уровней меню (от 0 до 9).

DRAFT

- Разделители не могут открывать подменю. Разделители могут отделять элементы одного уровня. Это значит, что разделитель не может быть первым, последним или единственным элементом в подменю.

Отображение Флажка для Элемента Меню

Когда элемент меню используется как переключатель, рядом с элементом отображается индикатор, информирующий о текущем состоянии системы.

Для этого в свойстве *SourceExpr* нужно указать выражение на языке C/AL, которое возвращает значение типа Boolean. Если возвращаемое значение равно TRUE, то возле элемента меню появится флажок.

11.6 Триггеры Форм и Элементов Управления

Среда C/SIDE распознает некоторые ситуации при использовании форм, такие как, открытие формы или нажатие на командную кнопку. При этом систему можно обязать выполнять определенный C/AL-код, указанный в соответствующих триггерах. Триггеры – это предопределенные функции, которые выполняются при наступлении некоторых событий. Изначально это функции пусты. Добавление C/AL кода в триггеры позволяет изменять поведение системы, так чтобы оно соответствовало поставленным задачам.

Обзор Триггеров Форм

К триггерам форм C/SIDE относятся:

Название Триггера Формы	Когда выполняется
OnInit	Форма загружена, но элементы управления не доступны
OnOpenForm	Форма проинициализирована (элементы управления доступны).
OnQueryCloseForm	Форма готовится к закрытию. Вызывается перед триггером *OnCloseForm*. Если в триггере вернуть значение FALSE форма не будет закрыта. Предназначен для запроса подтверждения у пользователя: действительно ли он хочет закрыть форму.
OnCloseForm	Форма закрывается. Вызывается после триггера *OnQueryCloseForm*.
OnActivateForm	Когда форма становится активным окном.
OnDeactivateForm	Форма перестает быть активным окном
OnFindRecord	Форма открыта, запись получена. Также вызывается при переходе к первой или последней записи
OnNextRecord	Указывается, как система должна выполнять выбор следующей записи (например, пользователь нажал клавишу PAGEDOWN в карточной форме).
OnAfterGetRecord	Запись была получена из базы данных, но еще не отображена.
OnAfterGetCurrRecord	Получена текущая запись. В элементе управления Табличная зона триггер *OnAfterGetRecord* вызывается для всех отображаемых записей, а данный триггер только для текущих.
OnBeforePutRecord	Запись готовится к сохранению
OnNewRecord	Новая запись проинициализирована, но еще не отображена

DRAFT

Название Триггера Формы	Когда выполняется
OnInsertRecord	Новая запись готовится к вставке в таблицу
OnModifyRecord	Запись готовится к сохранению модификации
OnDeleteRecord	Запись готовится к удалению из таблицы
OnTimer	После триггера *OnOpenForm,* а также после интервала времени, указанного в свойстве *TimeInterval*.
OnCreatehyperlink	При создании URL для формы (например, при отправке по электронной почте или на рабочий стол)
OnHyperlink	При нажатии по гиперссылке, выполняется после триггера *OnInit*.

В таблице приведено краткое описание назначения каждого триггера. Детальную информацию об этих триггерах см. в онлайновой справке *C/SIDE Reference Guide*.

Примечание

Три триггера *OnInsertRecord*, *OnModifyRecord*, и *OnDeleteRecord* соответствуют триггерам на уровне таблицы. Если одновременно используются триггеры и на уровне формы, и на уровне таблицы, первыми будут выполнены триггеры формы.

Обзор Триггеров Элементов Управления

В зависимости от типа элемента управления изменяется количество доступных триггеров. Статические элементы управления и контейнеры вообще не имеют триггеров, а для текстовых зон доступны все триггеры.

Остальные содержащие данные элементы управления и контейнеры, имеют доступ к части триггеров. Элементы управления, на которые можно нажать (командные кнопки, элементы меню, выключатели), имеют специальный триггер для обработки этой ситуации. В следующей таблице приведены все возможные триггеры элементов управления. В правой колонке показано, к каким элементам применяется данный триггер.

Название Триггера Элемента Управления	Когда выполняется	Элементу управления
OnActivate	При активации элемента управления	1,2,3,4,5,6,7,8
OnDeactivate	При деактивации элемента управления	1,2,3,4,5,6,7,8
OnFormat	Перед отображением элемента управления	5
OnBeforeInput	Элемент управления выбран для ввода, но в него еще ничего не введено	5
OnInputChange	При вводе данных пользователем. Данный триггер срабатывает после каждого введенного символа	5
OnAfterInput	Пользователь завершил ввод	5
OnPush	Нажат элемент управления	1,3,4,6,7,9
OnValidate	Элемент управления теряет фокус	3,4,56,7
OnAfterValidate	Введенные данные была проверены.	3,4,5,6,7
OnLookup	Пользователь запускает функцию Выбор (щелкнул кнопку Выбор или нажал клавишу F6).	5

DRAFT

Название Триггера Элемента Управления	Когда выполняется	Элементу управления
OnDrillDown	Пользователь запускает функцию Детализировать	5
OnAssistEdit	Пользователь запускает функцию assist-edit	5

Элементу управления

1 – командная кнопка, 2 – кнопка меню, 3 - выключатель, 4 – кнопка выбора, 5 – текстовая зона, 6 – зона изображения, 7 - индикатор, 8 – сопутствующая форма, 9 - элемент меню

Подробное описание данных триггеров см. в онлайновой справке *C/SIDE Reference Guide*.

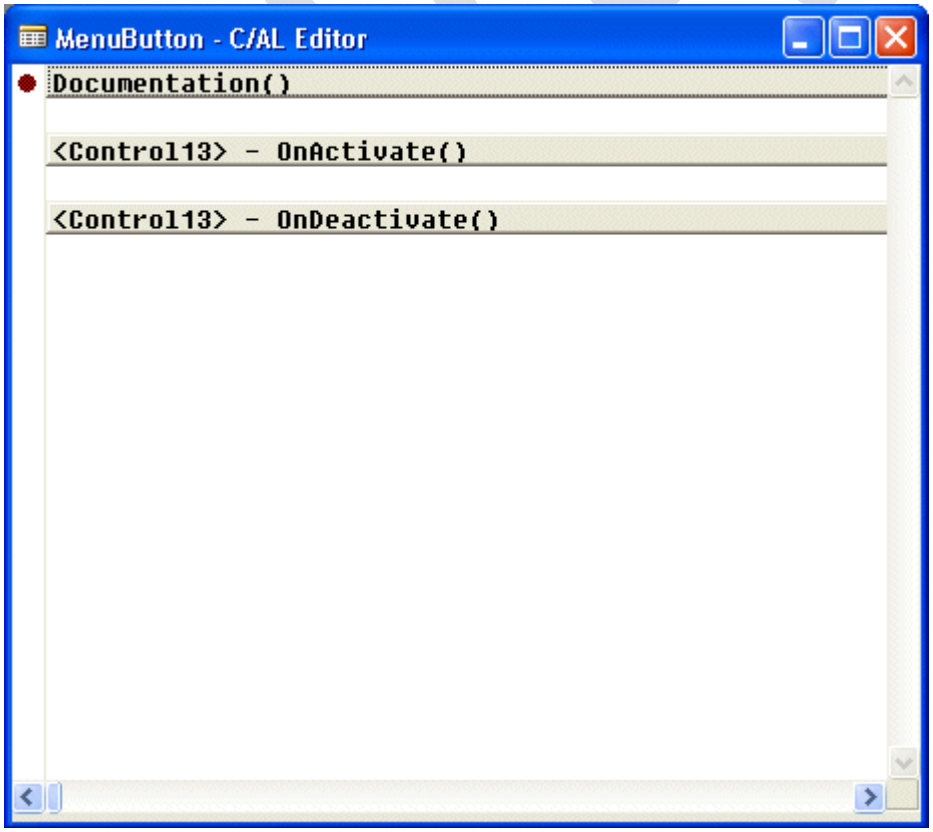
> **Примечание**
>
> На уровне поля таблицы также есть триггер *OnValidate*. Если определены оба триггера (для поля и для элемента управления), триггер поля выполняется перед триггером элемента управления (системные проверки выполняются самыми первыми).
>
> *OnLookup* также есть на уровне таблицы. Однако в данном случае очередность выполнения другая. Триггер на уровне формы имеет преимущество перед триггером на уровне таблицы. Оба триггера имеют преимущество перед системной функцией Выбор, используемой по умолчанию.

Как Создавать и Изменять Триггеры Формы и Элемента Управления

Чтобы создать или изменить триггер формы или элемента управления нужно:

1. Открыть форму в **Form Designer**.

2. Выбрать форму или элемент управления (элемент меню) и щелкнуть Вид, C/AL Code (F9) чтобы открыть окно ***C/AL Editor***:



3. В редакторе доступны триггеры, которые могут использоваться для выбранного объекта. Здесь можно добавить или изменить C/AL код в подходящих триггерах.

4. Щелкните Сервис, Compile (F11) чтобы выполнить тестовую компиляцию формы.

Если вы не знакомы с программированием на языке C/AL, ознакомьтесь с главой ГЛАВА 16 ВВЕДЕНИЕ В ЯЗЫК C/AL.

11.7 Типы и Характеристики Форм

В этом разделе описываются типы окон, которые пользователь видит в Dynamics NAV. Затем обсуждаются типы форм, которые вы будете создавать в своих приложениях, а также даются примеры и характеристики каждого из них.

В Dynamics NAV существует пять основных видов окон.

Диалоги (dialogs)

Это простые окна, отображающие информацию, работа с которыми заключается в нажатии кнопок, таких как OK, Cancel, Yes, No и др. Они генерируются C/AL-функциями, такими как ERROR, MESSAGE, TESTFIELD, CONFIRM и так далее, или создаются программистом, использующим переменную типа Dialog. Они не связаны с объектами форм.

Формы запросов (request forms)

Они находятся только в объектах отчетов и не связаны с объектами форм.

Несвязанные формы (unbound forms)

Это объекты форм, не связанные с таблицей. Как и панели запросов, они используются, когда для продолжения обработки данных необходимо задать пользователю несколько вопросов.

Формы с одной записью (one-record forms)

Это объекты форм, связанные с таблицей. Они позволяют пользователю видеть и, возможно, редактировать только по одной записи из одной таблицы. В качестве примера можно привести карточные формы.

Формы с несколькими записями (multi-record forms)

Это объекты форм, связанные с таблицей. Они позволяют пользователю одновременно видеть несколько записей из одной таблицы и, возможно, редактировать их. В качестве примеров можно привести табличные (Tabular Forms), трендовые (TrendScape Forms) и матричные (Matrix Forms) формы.

Формы, связанные с таблицами (последние две в списке) являются строительными блоками системы Dynamics NAV. Хотя среда C/SIDE позволяет вам создавать множество типов форм с различным оформлением и методами функционирования, приложение Dynamics NAV использует только некоторые из них. Это обеспечивает единство оформления и стиля приложения.

Настоятельно рекомендуется в целях облегчения использования и единства стиля соблюдать эту политику при модификации системы Dynamics NAV и любого приложения, которое взаимодействует с ней. В последующем описании полностью учтены стандарты системы Dynamics NAV.

Типы Форм и Примеры

В системе Dynamics NAV существует несколько стандартных типов форм. Ниже приведена таблица с наиболее часто используемыми типами форм и несколькими примерами каждой из них. Более подробное описание каждого типа дано в следующих разделах.

Тип	Одна запись/ Несколько записей	Примеры

DRAFT

Тип	Одна запись/ Несколько записей	Примеры
Карточная Форма (Card Form)	Одна запись	Customer Card, Vendor Card, Item Card
Форма Статистики (Statistics Form)	Одна запись	Customer Statistics, Vendor Statistics, Employee Statistics
Табличная Форма (Tabular Form)	Несколько записей	Currencies, Payment Terms
Списочная Форма (List Form)	Несколько записей	Customer List, Item List, Item Ledger Entries
Рабочая Форма (Worksheet Form)	Несколько записей	General Journal, Cash Receipts Journal, Item Transfer Journal
Форма заголовка и форма строк (Header Form,Line Form)	Оба варианта (фактически две формы)	Sales Invoice, Posted Purchase Credit Memo, Finance Charge Memo
Настроечная Форма (Setup Form)	Одна запись	General Ledger Setup, Company Information, Sales & Receivables Setup

Понимание этих стандартов, даст вам возможность понимать области приложения, бросив беглый взгляд на объекты

Карточная Форма (Card Form)

Карточная форма позволяет пользователю просматривать и редактировать по одной записи в таблице. Она используется, когда существует слишком много полей, чтобы можно было просматривать их, используя только одну строку. Карточные формы всегда содержат закладки у верхнего края, благодаря которым пользователь может выбирать и отображать различные группы полей.

Даже если полей немного, существует, по крайней мере, одна вкладка Общее (General), которая всегда стоит первой.

Поле первичного ключа (Primary Key) таблицы – всегда первое поле на вкладке Общее. В таблицах, где используются карточные формы, имеется только одно поле в первичном ключе.

Присвоение имен карточным формам

Картотечные формы получают имя таблицы, с которой они связаны, за ним следует слово «Card».

Например, карточная форма, связанная с таблицей Customer, называется **Customer Card**. Картотечные формы также имеют, по крайней мере, одну кнопку меню в своей нижней части, совпадающую с именем таблицы, и обеспечивает доступ к соответствующей информации.

Форма Статистики (Statistics Form)

Форма статистики – форма для одной записи, которая позволяет пользователю просматривать данные, но не редактировать их. Обычно она содержит поля FlowFields, которые позволяет пользователю получать дополнительную информацию. Обычно она также содержит вычисляемые или производные данные, содержащиеся в переменных, которые получить напрямую.

Кроме того, статистические формы могут содержать вкладки, способствующие организации данных.

Первичный ключ таблицы отображается только в строке заголовка формы.

Присвоение имен формам статистики

DRAFT

Формы статистики получают имя таблицы, с которой они связаны, за ним следует слово «Statistics».

Например, статистическая форма, связанная с таблицей **Customer** называется **Customer Statistics**.,

Формы статистики операций (Entry Statistics) – это специальная версия формы статистики. Они получает имя таблицы, с которой связаны, за ним следует словосочетание «Entry Statistics». Например, **Customer Entry Statistics**.

Табличная Форма (Tabular Form)

Табличная форма – это форма для нескольких записей, позволяющая пользователю просматривать и редактировать ряд записей из таблицы. Каждая запись отображается как отдельная строка в табличной форме, а каждое поле отображается как столбец, в результате чего непосредственно на форме формируется таблица, похожая на электронную таблицу.

Первичный ключ связанной таблицы отображается в крайнем левом столбце. Если в первичном ключе содержится несколько полей, они отображаются в порядке их значимости в столбцах, начиная слева.

Присвоение имен табличным формам

Табличные формы получают имя таблицы, с которой они связаны, с использованием множественного числа.

Например, табличная форма, связанная с таблицей ***Country/Region*** называется ***Countries/Region***.,

Если связанные таблицы имеют несколько полей в первичном ключе, имя может быть иным. Например, табличная форма, связанная с таблицей **General Posting Setup**, называется **General Posting Setup**.

Списочная Форма (List Form)

Списочная форма – это форма для нескольких записей, позволяющая пользователю просматривать ряд записей из таблицы, но не редактировать их. В ней, как и в табличной форме, используются строки и столбцы.

Поле (поля) первичного ключа связанной таблицы отображается в крайнем левом столбце.

Присвоение имен списочным формам

Табличные формы получают имя таблицы, с которой они связаны, за ним следует слово «List». Например, списочная форма, связанная с таблицей **Customer** называется **Customer List**.

«Специализированная» форма книги (Ledger)

Более специализированная версия списочной формы – это форма книги (Ledger). Она используется только для таблиц Ledger Entry. Эта форма отличается от обычных списочных форм тем, что хотя пользователи не могут вставлять или удалять записи, им разрешено иногда редактировать небольшое количество полей.

Кроме того, первичный ключ, всегда являющийся целым числом с именем «Entry No.», отображается чаще в крайнем правом, а не крайнем левом столбце.

И, наконец, форма книги получает имя связанной таблицы во множественном числе. Например, форма книги, связанная с таблицей **Customer Ledger Entry**, называется **Customer Ledger Entries**.

Рабочая Форма (Worksheet Form)

Рабочая форма – это особая версия табличной формы для нескольких записей, позволяющая пользователю просматривать несколько записей из таблицы и редактировать их. Различие заключается в том, что когда пользователь вставляет новую запись, эта запись не переходит в другую позицию на форме, а остается в том же порядке, в каком пользователь ввел ее.

DRAFT

Подобный эффект достигается с помощью свойства AutoSplitKey формы, в сочетании с целым числом в качестве последнего поля в первичном ключе таблицы.

Поля первичного ключа связанной таблицы не отображаются на рабочей форме.

Присвоение имен рабочим формам

Рабочие формы названы в соответствии с назначением связанной таблицы, что рассматривается ниже в этом разделе. В качестве примера можно привести Journal Table. В данном случае имя рабочей формы завершится словом «Journal».

Форма Заголовка и Форма Строк (Header/Line Forms)

Многие формы в системе Dynamics NAV обладают характеристиками и картотечной формы, и табличной формы, в частности, форма Sales Invoice.

Поля, которые являются общими для всего счета-фактуры, расположены на форме наподобие карточек с вкладками, отображая один счет-фактуру. Однако строки счета отображаются в разделе формы, напоминающей таблицу, где одновременно несколько строк счета-фактуры (из того же счета-фактуры) можно просматривать и редактировать. Это фактически две отдельные формы, связанные с двумя различными таблицами. Главная форма – это карточная форма, связанная с таблицей. Главная форма содержит сопутствующую форму. Рабочая форма отображается в сопутствующей форме, связанной с другой таблицей, которая, в свою очередь, является «вспомогательной» для первой формы. Элемент управления Сопутствующая форма отвечает за взаимосвязь между этими двумя формами.

Присвоение имен формам заголовка и строк

Во многих случаях эти две формы вместе представляют документ.

В приведенном выше примере – форме Sales Invoice – именем формы будет имя документа, который она представляет. В других случаях имя формы будет таким, каким было бы имя главной формы без сопутствующей формы.

Настроечная Форма (Setup Form)

Установочная форма – форма с одной записью, которая позволяет пользователю просматривать и редактировать одну и только одну запись в таблице установки. При использовании данной формы пользователю не разрешается вставлять или удалять эту запись. В форме имеется много полей, поэтому они содержат вкладки у верхнего края для организации данных. Поскольку там есть только одна запись, первичный ключ не отображается на этой форме.

Присвоение имен настроечным формам

Установочные формы носят имя таблицы, с которой они связаны. Например, установочная форма, связанная с таблицей **General Ledger Setup**, называется **General Ledger Setup**.

Другие Формы с Несколькими Записями

Существуют другие формы, которые позволяют пользователям просматривать и/или редактировать одновременно несколько записей. В их число входят формы трендовые (TrendScape) и матричные (Matrix). Эти формы предоставляют дополнительную функциональность в Dynamics NAV, но не оказывают никакого влияния на архитектуру.

Пример трендовой формы (TrendScape) - ***Contract Trendscape***.

Пример матричной формы (Matrix) - ***Budget***.

Стандартные Средства Навигации

Материал этого раздела покажет способы перехода от одной формы к другой. Здесь мы просто упомянем о стандартных средствах навигации, которые обеспечивают единство подходов во всей системе.

DRAFT

Карточные и Списочные Формы

Для большинства главных таблиц существуют оба вида форм: карточная и списочная. Поскольку обе эти формы обращаются к одной таблице, существует много стандартов, определяющих действия каждой формы.

- В карточной форме, пользователь должен просматривать и редактировать по одной записи.
- Пользователь должен иметь возможность просматривать (но не редактировать) все записи в списочной форме. Так как эта форма недоступна для редактирования, пользователь может начать поиск, просто вводя текст в определенном столбце.
- В карточной форме должна быть возможность вызвать списочную форму следующими способами:

  Клавиша F5.

  Пункт меню Список (в меню кнопки, с таким же названием, как у таблицы).

  Кнопка Список на панели инструментов.
- В форме списка пользователь должен иметь возможность выбрать новую запись и щелкнуть кнопку OK, чтобы закрыть форму списка и вернуться в карточную форму. После этого карточная форма должна отображать запись, выбранную в форме списка.
- Пользователь должен иметь возможность открыть новую карточную форму, нажав клавиши SHIFT+F5 и выбрав пункт меню Карточка (в меню кнопки, с таким же названием, как у таблицы). При этом должна появиться карточная форма, отображающая выбранную запись.

Формы Статистики

Как сказано выше, большинство главных таблиц содержат как карточную форму, так и списочную форму. Кроме того, многие главные таблицы содержат форму статистики. Пользователь должен иметь возможность получать доступ к этой форме либо из карточной формы, либо из списочной формы одним и тем же способом.

Стандартный способ получения доступа к статистической форме – с помощью клавиш быстрого доступа F9 или пункта меню Статистика (на кнопке меню, у которой такое же имя, что у таблицы).

Формы Книги

Каждая главная таблица содержит, по меньшей мере, одну (обычно единственную) книгу. Пользователь должен иметь возможность перейти из главных форм (карточной, списочной и статистики) в книгу. Он может сделать это тремя различными способами: используя кнопку Детализировать в поле FlowField, применив сочетание клавиш CTRL+F5 или выбрав пункт меню «Книга Операций» (Ledger Entries) (нажав на кнопку меню, имеющую то же имя, что у таблицы).

При нажатии кнопки Детализировать на экран выводятся только те записи, которые используются для вычисления значения (это функция, входящая во встроенные функциональные возможности). Если используется сочетание клавиш или пункт меню, будут показаны все записи книги операций данной главной таблицы. Для этого требуется использовать ссылку RunFormLink в пункте меню, связывающую данные две таблицы. При каждом обновлении главной формы потребуется также синхронизация этих двух форм. Для этого необходимо установить значение OnUpdate в свойстве RunFormLinkType.

Примечание

В целях производительности, если используется свойство *RunFormLink,* следует устанавливать соответствующее значение в свойстве *RunFormView.* Сортировка, указанная в свойстве *RunFormView,* должна содержать поля, перечисленные в свойстве *RunFormLink,* иначе производительность ухудшится.

DRAFT

Форма Журнала

Каждая форма журнала содержит схожие кнопки в нижней части формы. Эти кнопки позволяют пользователю либо переходить к другим формам, либо выполнять какие-либо действия.

Обычно на форме журнала расположены следующие кнопки: Строка, Функции, Учет, кнопка меню, имеющая то же имя, что и главная таблица функциональной области. Эта кнопка меню должна содержать пункты меню, позволяющие пользователю переходить прямо в карточную форму или в книгу операций. В обоих случаях выведенная форма должна содержать ссылку на запись главной таблицы, с которой связана данная строка книги учета.

Кнопка меню с именем “Учет” должна содержать следующие пункты меню, относящиеся к записи строк в книгу учета: Учет, Учет и Печать, Тестовый Отчет

DRAFT

ЧАСТЬ 5 ОТЧЕТЫ

ГЛАВА 12 ОТЧЕТЫ – БАЗОВЫЕ ПОНЯТИЯ

Отчеты используются для вывода на печать информации из базы данных. Отчет может использоваться для структурирования и суммирования информации, а также для вывода на печать документов, таких как счет-фактура. Кроме того, отчеты могут использоваться для обработки данных без вывода их на печать.

В этой главе представлены фундаментальные принципы и основные задачи, которые необходимо учитывать при конструировании отчетов.

- Что такое отчет?
- Что происходит при запуске отчета?
- Report Designer
- Сохранение, компиляция и запуск отчетов

12.1 Что Такое Отчет?

Отчеты в Dynamics NAV служат нескольких целям.

- Отчеты используются для вывода в структурированном виде на печать информации из базы данных. Например, можно создать отчет, содержащий список всех клиентов, с указанием всех заказов, сделанных каждым клиентом.
- Все создаваемые документы, относящиеся к приложению, должны быть сделаны в виде отчета. Например, чтобы напечатать счет-фактуру, требуется создать отчет, в который автоматически будет внесена необходимая информация.
- Отчеты могут не выводиться на печать. Это может показаться противоречием, но на самом деле это не так. Отчет может использоваться для автоматизации множества повторяющихся задач, таких как обновление всех цен в списке товаров. Это можно сделать внесением C/AL-кода в программный модуль, но использование отчета значительно облегчает эту задачу, потому что вы можете использовать мощные средства моделирования данных, предназначенные для формирования отчета.

Компоненты Отчета

На следующей схеме представлены компоненты отчета и их взаимоотношения:

DRAFT

Описание отчета

Это полное описание отчета (report description) – того, как данные собираются и представляются на бумаге при выполнении отчета. Описание отчета хранится в базе данных.

Элемент данных

Элемент данных (data item) относится к таблице. Чтобы извлекать информацию из таблиц в базе данных, вы определяете элементы данных. Когда в отчете используется несколько таблиц, вы устанавливаете отношения между элементами данных, чтобы извлекать и организовывать данные нужным образом.

Раздел

В печатном отчете каждый элемент данных может содержать один или несколько разделов. Раздел (section) можно представить себе как блок информации, печатаемой на бумаге. Законченный отчет состоит из ряда разделов. Некоторые из них, такие как итоговая сумма, выводятся на печать только один раз. Другие, например заголовок, выводятся на печать один раз на каждой странице. Третьи печатаются каждый раз при извлечении новой записи из базы данных.

Элемент управления

Информация, которая печатается в разделах, состоит из элементов управления (controls). Имеющиеся элементы управления: текстовые зоны (text boxes) предназначены для вывода на печать результата вычисления любого допустимого C/AL-выражения, например содержания поля таблицы (а также сложных вычислений); надписи (labels) для вывода на печать статического текста, например, заголовка столбца данных; а фигуры (shapes), изображения (images) и зоны изображения (picture boxes) предназначены для вывода на печать графических элементов (линий, кругов) и растровых изображений в отчете.

Форма запроса

Форма запроса (request form) – это форма, которая исполняется до того, как фактически начинает выполняться отчет. Она используется для получения запросов и вариантов выбора от пользователя - например, порядка сортировки или уровня детализации.

DRAFT

Свойство

Свойство (property) – это атрибут объекта (отчета, элемента данных, раздела и т.д.), который некоторым образом характеризует объект (его цвет, размер, отображается ли он и многое другое). Свойства устанавливаются в окне **Property** данного объекта.

Триггер

Некоторые предопределенные события, которые происходят с отчетом, позволяют системе выполнять задаваемую пользователем C/AL-функцию: событие запускает функцию. Как видно из диаграммы, сам отчет, элементы данных, разделы, форма запроса и элементы управления, имеющиеся на форме запроса, содержат триггеры.

Логическое и Визуальное Конструирование

В конструировании отчета есть две стороны:

- определение логической структуры или модели данных
- разработка визуальной компоновки.

Определение модели данных – это определение того, как собираются данные для отчета. Сюда входит следующее:

- определение элементов данных, в которых указываются таблицы, используемые отчетом;
- определение отношений между элементами данных, если отчет использует несколько таблиц;
- определение ключа, порядка сортировки и фильтров, которые будут использоваться с соответствующими элементами данных;
- определение того, как должны быть сгруппированы данные;
- определение того, как должны рассчитываться промежуточные и общие суммы;
- возможно, написание C/AL-кода в триггерах элементов данных для улучшения функциональных возможностей.

Элементы Данных (Data items)

Модель данных отчета строится на основе элементов данных. Элемент данных относится к таблице. При выполнении отчета, каждый элемент данных повторяется для всех записей в соответствующей таблице. Когда отчет основан на нескольких таблицах, для управления сбором информации устанавливается иерархия элементов данных, для этого задаются отступы для элементов данных.

> **Пример**
>
> Чтобы создать отчет, при котором на печать выводится список клиентов, а для каждого клиента перечислены заказы на покупку, размещенные этим клиентом, вы должны определить два элемента данных. Один, который соответствует таблице **Customer,** и второй, который соответствует таблице **Sales Order**. Для второго элемента данных задан отступ (indentation) – поскольку в отчете собираются данные из записей в таблице **Customer,** для каждого клиента все заказы на покупку, которые относятся к этому клиенту, должны быть найдены посредством считывания записей в таблице **Sales Order**.

Разделы (Sections)

Визуальная компоновка отчета предполагает наличие разделов. В печатном отчете (помните, что отчеты не обязательно печатают что-либо) к каждому элементу данных может быть присоединен один или несколько разделов. Существует несколько типов разделов, каждый из которых обладает определенной функцией. Обычно большая часть данных выводится на печать в разделе области данных (body section) элемента данных, тогда как раздел заголовка элемента данных используется для вывода на печать информации до того, как какая-либо запись элемента данных будет

DRAFT

напечатана (например, заголовки столбцов). Однако существуют отчеты, где раздел области данных не используется вообще, и вся информация печатается в других разделах.

На следующем рисунке показан законченный отчет.

Sales Statistics
CRONUS International Ltd.

No.	Name	_before	25-01-01 24-02-01	25-02-01 24-03-01	25-03-01 24-04-01	after_
01445544	**Progressive Home Furnis**					
	Sales (LCY)	1,499.03	0.00	0.00	0.00	0.00
	Profit (LCY)	305.13	0.00	0.00	0.00	0.00
	Profit %	20.4	0.0	0.0	0.0	0.0
	Inv. Discounts (LCY)	0.00	0.00	0.00	0.00	0.00
	Pmt. Discounts (LCY)	0.00	0.00	0.00	0.00	0.00
	Pmt. Disc Tol. (LCY)	0.00	0.00	0.00	0.00	0.00
	Pmt. Tolerances (LCY)	0.00	0.00	0.00	0.00	0.00
01454545	**New Concepts Furniture**					
	Sales (LCY)	222,241.32	0.00	0.00	0.00	0.00
	Profit (LCY)	0.00	0.00	0.00	0.00	0.00
	Profit %	0.0	0.0	0.0	0.0	0.0
	Inv. Discounts (LCY)	0.00	0.00	0.00	0.00	0.00
	Pmt. Discounts (LCY)	0.00	0.00	0.00	0.00	0.00
	Pmt. Disc Tol. (LCY)	0.00	0.00	0.00	0.00	0.00
	Pmt. Tolerances (LCY)	0.00	0.00	0.00	0.00	0.00
10000	**The Cannon Group PLC**					
	Sales (LCY)	309,122.34	3,150.00	0.00	0.00	0.00
	Profit (LCY)	5,385.46	2,419.89	0.00	0.00	0.00
	Profit %	1.7	76.8	0.0	0.0	0.0
	Inv. Discounts (LCY)	727.34	0.00	0.00	0.00	0.00
	Pmt. Discounts (LCY)	0.00	0.00	0.00	0.00	0.00
	Pmt. Disc Tol. (LCY)	0.00	0.00	0.00	0.00	0.00
	Pmt. Tolerances (LCY)	0.00	0.00	0.00	0.00	0.00
20000	**Selangorian Ltd.**					
	Sales (LCY)	218,087.89	0.00	0.00	0.00	0.00
	Profit (LCY)	2,511.90	0.00	0.00	0.00	0.00
	Profit %	1.2	0.0	0.0	0.0	0.0
	Inv. Discounts (LCY)	201.35	0.00	0.00	0.00	0.00

Этот отчет содержит статистическую информацию о продажах, извлеченную из одной таблицы. Он демонстрирует ряд особенностей, которые доступны при конструировании отчетов.

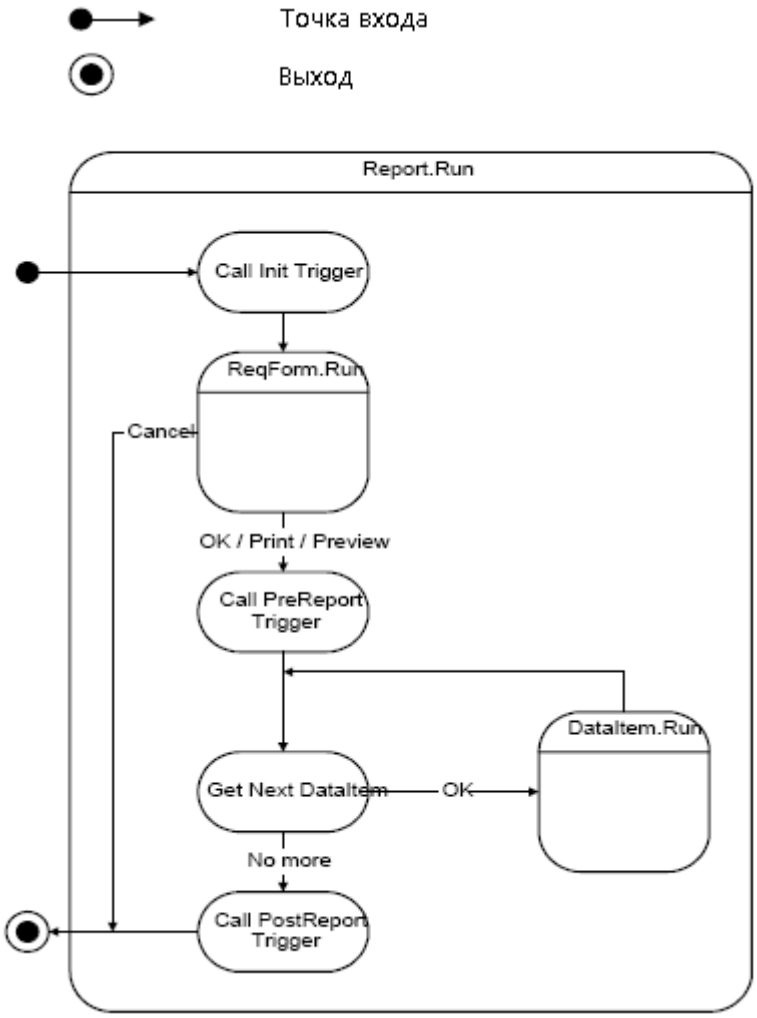
- Перед выводом на печать какой-либо записи из таблицы дается заголовок (header), содержащий название и информацию о фильтре, который использовался по отношению к номерам клиентов.
- Каждый раздел области данных содержит информацию о клиенте, размещенную на нескольких строках. Строки «%» вычисляются при выполнении отчета.
- После того как все записи (т.е. записи, выбранные имеющимся фильтром) напечатаны, выводится раздел нижнего колонтитула, содержащий общую сумму для выбранных клиентов.
- В разделе области данных и в разделе нижнего колонтитула использован фильтр для создания столбцов, где собраны данные и вычислена их сумма за разные периоды времени.

12.2 Что Происходит при Запуске Отчета?

Следующая схема является упрощенной версией схемы из Приложения A Report Flow Charts. Если вы хотите ознакомиться со всеми деталями, включая последовательность работы триггеров и их назначение см. приложение. В данном разделе приведено описание работы отчета в общих чертах.

Запуск Отчета

На первой схеме приведены события, которые происходят при запуске отчета:



1. При инициализации отчета вызывается триггер *OnInitReport*. В этом триггере выполняются операции, которые должны быть сделаны до того, как отчет будет запущен. Здесь же можно отменить открытие отчета.

2. Если триггер *OnInitReport* не остановил выполнение отчета, запускается форма запроса отчета (если она создана). Здесь можно указать параметры, с которыми должен быть запущен отчет. Также на этом шаге можно отменить отчет.

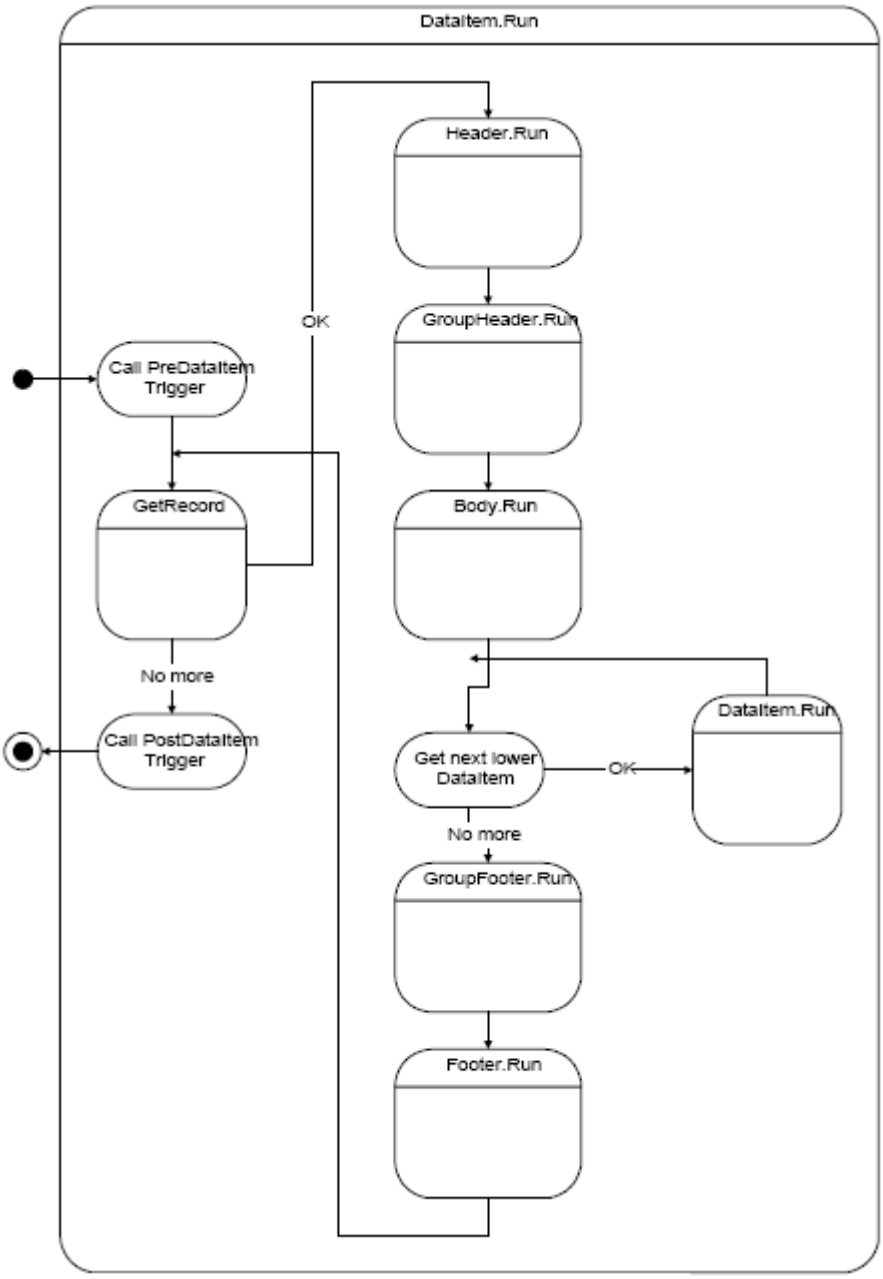
3. Если было решено продолжить, вызывается триггер *OnPreReport*. В этот момент никакие данные еще не были обработаны.

4. После работы триггера *OnPreReport* обрабатывается первый элементы данных (предполагается, что отчет не был завершен в триггере *OnPreReport*).

5. После того как была завершена обработка первого элемента данных, система приступает к обработке второго элемента данных (эта обработка выполняется точно так же).

6. Когда больше не осталось необработанных элементов данных вызывается триггер *OnPostReport,* в котором выполняются необходимые завершающие процедуры (например, удаление временных файлов).

На следующей схеме детально раскрываются процедуры, выполняющиеся при обработке элемента данных (шаг 4):



1. Перед тем как будет получена первая запись, вызывается триггер *OnPreDataItem,* а после обработки последней записи вызывается триггер *OnPostDataItem*.

2. Между этими двумя триггерами выполняется обработка записей. Обработка записей подразумевает вызов триггеров записей и вывод разделов. Среда C/SIDE также определяется должна ли текущая запись вызывать вывод специальных разделов, таких как header, footer, group header или group footer.

3. Если есть вложенные элементы данных, то для них выполняется аналогичная процедура (разрешается 10 уровней вложенности)

4. Когда в элементе данные больше не остается записей, управление возвращается к той точке, откуда был инициирован процесс. Для вложенных элементов данных – это следующая запись

элемента данных более высокого уровня. Если нет элементов, данных более высокого уровня (вложенность 0) управление передается в отчет (далее как показано на предыдущей схеме).

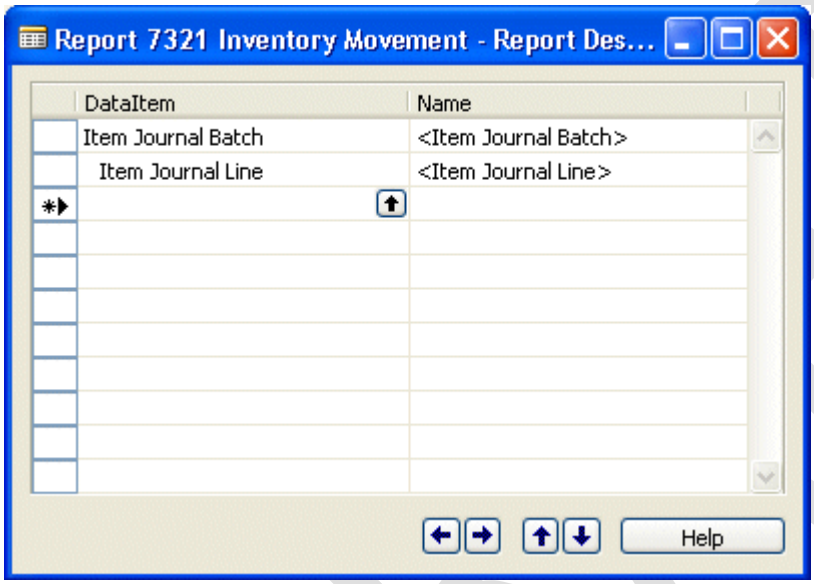
12.3 Report Designer

Report Designer используется для создания отчетов, необходимых в разрабатываемом приложении. Report Designer содержит два дополнительных дизайнера:

- Section Designer используется для визуальной компоновки отчетов
- Request Options Form Designer используется для создания форм запросов отчета.

Report Designer

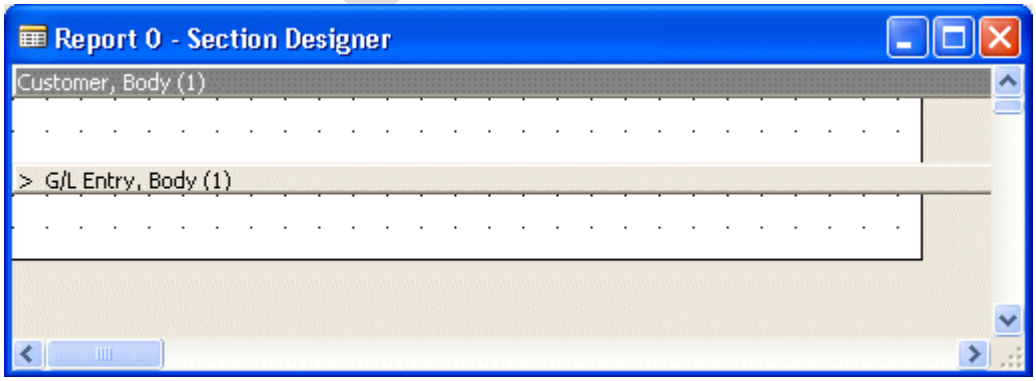
Окно ***Report Designer*** позволяет определять модель данных, добавляя элементу данных и указывая их вложенность:



Для каждого элемента данным можно редактировать свойства и триггера, для этого нужно соответственно открывать окно ***Properties*** или C/AL Editor. Для редактирования свойств самого отчета выберите пустую строку в ***Report Designer,*** и откройте окно ***Properties***. Чтобы редактировать триггеры самого отчета выберите пустую строку в ***Report Designer,*** и щелкните Вид, C/AL Code. Как вариант можно щелкнуть Правка, Выбрать Объект, а затем открыть окно ***Properties*** или C/AL Editor.

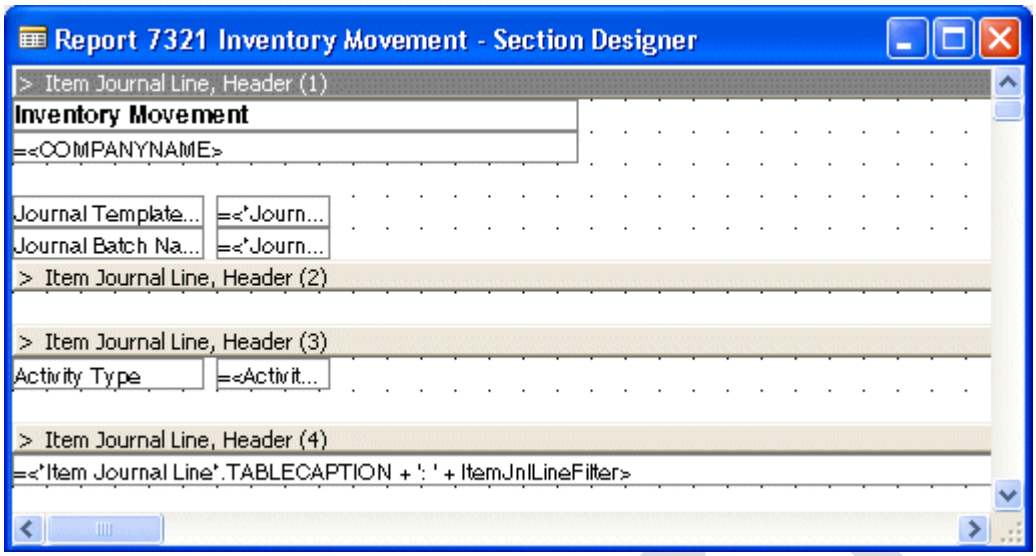
Section Designer

После того как был создан один или несколько элементов данных, можно приступать к конструированию визуальной компоновка отчета:



Для выбора и вставки полей в разделы в качестве элементов управления можно использовать **Field Menu**. Оно было описано в части книги, посвященной формам.

На следующем рисунке, некоторое количество текстовых зон (text box) и надписей (label) было помещено в четырех разделах.



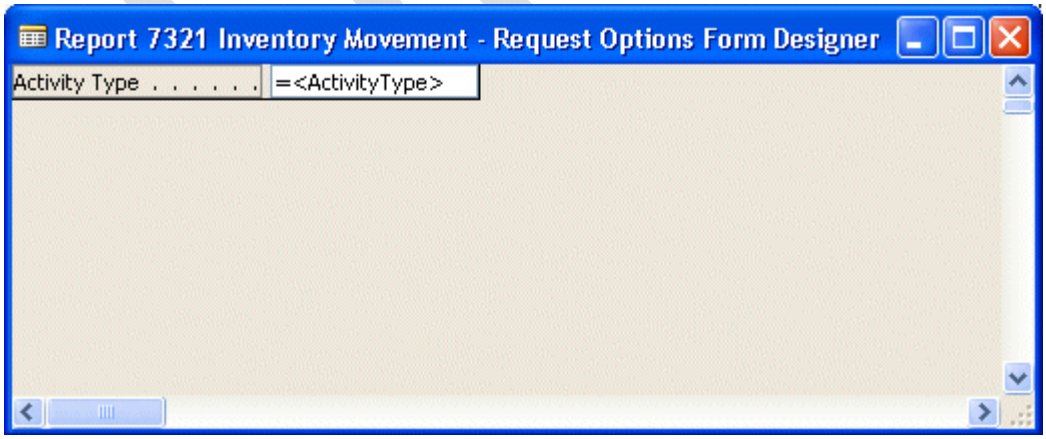
Каждый раздел можно представлять как одну или несколько строк, которые будут в итоге напечатаны отчетом. Раздел header (заголовок) будет напечатан только один раз, а раздел body обычно печатается столько раз, сколько циклов было выполнено в отчете. Можно указать, чтобы раздел header печатался на каждой странице.

Для каждого раздела и можно редактировать свойства и триггеры, открывая соответственно окно ***Properties*** или C/AL editor для выбранного раздела.

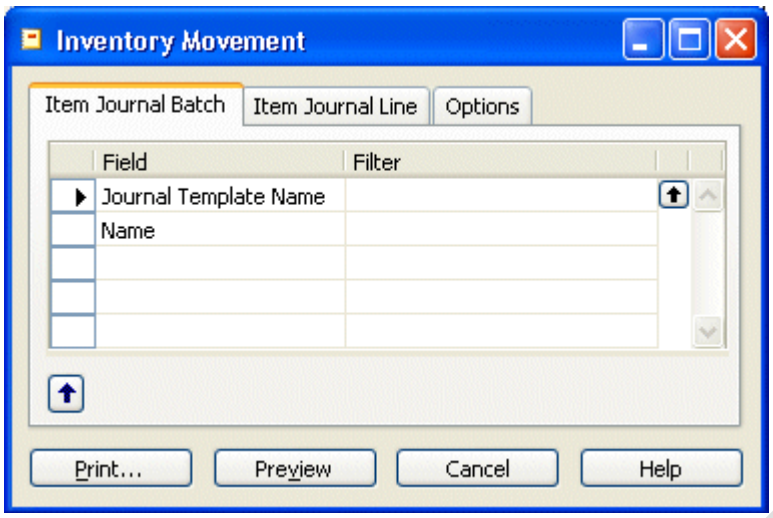
Для элементов управления также есть набор свойств, аналогичный тому который используется для элементов управления на формах (не все свойства могут использоваться в отчетах). Для изменения свойств можно использовать те же инструменты (Font Tool, Color Tool). Для просмотра свойств используется окно ***Properties.***

Request Options Form Designer

Для создания формы, которая будет запрашивать у пользователя дополнительные параметры перед формированием отчета, используется Request Options Form Designer. Данный дизайнер работает так же как Form Designer:



Request Options Form Designer используется, если пользователь перед формированием отчета должен выбрать ряд параметров. Когда отчет запускается, форма запроса выглядит следующим образом:



Как можно видеть, создается форма с закладками. Первые две закладки относятся к элементам данных. Они создаются автоматически (их содержимым можно управлять с помощью свойств элементов данных) и используются для установки фильтров и порядка сортировки.

Последняя закладка **Параметры** появляется, если была создана форма запроса с помощью **Request Options Form Designer**.

Свойства и триггеры формы запроса такие же, как у любой другой формы, также на нее можно поместить любые элементы управления.

12.4 Сохранение, Компиляция и Запуск Отчетов

После того как отчет был создан, его требуется сохранить и откомпилировать, перед тем как его можно будет запускать. Обычно это делается после окончания работы над отчетом. Однако может потребоваться сохраниться отчет работа, над которым еще не завершена, и который не может быть откомпилирован, например, из-за незаконченного кода на языке C/AL. Также возможно выполнить тестовую компиляцию не закрывая или сохраняя отчет.

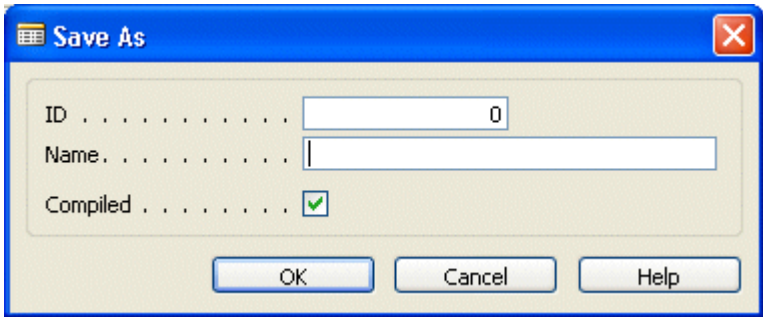
Сохранение и Компиляция Отчета

Отчет закрывается при закрытии окна ***Report Designer***.

Чтобы сохранить отчет:

1. При закрытии отчета, C/SIDE спросит, требуется ли сохранить отчет. Если это новый отчет (отчет, который до этого не сохраняли) потребуется задать для него идентификатор и наименование. Идентификатор должен быть уникальным и соответствовать правилам нумерации объектов – ваш сертифицированный партнер Microsoft с компетенцией Business Solutions предоставит вам эту информацию.

Если вы ввели идентификатор (ID) и наименование (Name) в свойствах отчета, система использует эти значения и их не потребуется вводить при закрытии отчета.



DRAFT

2. Параметр **Compiled** выбран по умолчанию (установлен флаг). Если ваш отчет не может быть скомпилирован, уберите флаг из данного поля.

3. Щелкните кнопку OK, чтобы сохранить отчет.

Отчет можно сохранить, не закрывая его, для этого щелкните Файл, Save или Save As... Вы можете использовать Save As... чтобы дать отчету новое наименование.

Отчет, как и другие объекты C/SIDE, должен быть скомпилирован, перед тем как его можно будет запускать. Как было описано ранее, компиляцию отчета можно выполнять при сохранении.

В процессе работы над отчетом может потребоваться тестовая компиляция, чтобы найти возможные ошибки. Это особенно полезно, когда отчет содержит C/AL код в триггерах. Тестовую компиляцию можно запустить, щелкнув Сервис, Компилировать.

Запуск Отчета

В полностью завершенном приложении отчеты интегрированы в меню или вызываются из других форм. Однако при создании отчета, часто необходимо запустить отчет до того как он будет интегрирован в приложение.

Тестовый запуск отчета

При работе над отчетом можно выполнить тестовый запуск, щелкнув для этого Файл, Run. Текущая версия отчета будет откомпилирована и запущена. При этом отчет не сохраняется, поэтому данная возможность может использоваться для проверки работоспособности внесенных изменений, до того как они будут сохранены.

Запуск отчета из Object Designer

Вы можете запустить отчет из списка отчетов, выбрав его в Object Designer и щелкнув кнопку Run.

DRAFT

ГЛАВА 13 РЕДАКТИРОВАНИЕ ОТЧЕТОВ

В данной главе приведено описание свойств отчетов, также в главу включено два примера, демонстрирующих базовые шаги, касающиеся редактирования отчетов.

- Свойства отчета
- Создание простого отчета
- Создание более сложного отчета

13.1 Свойства Отчетов

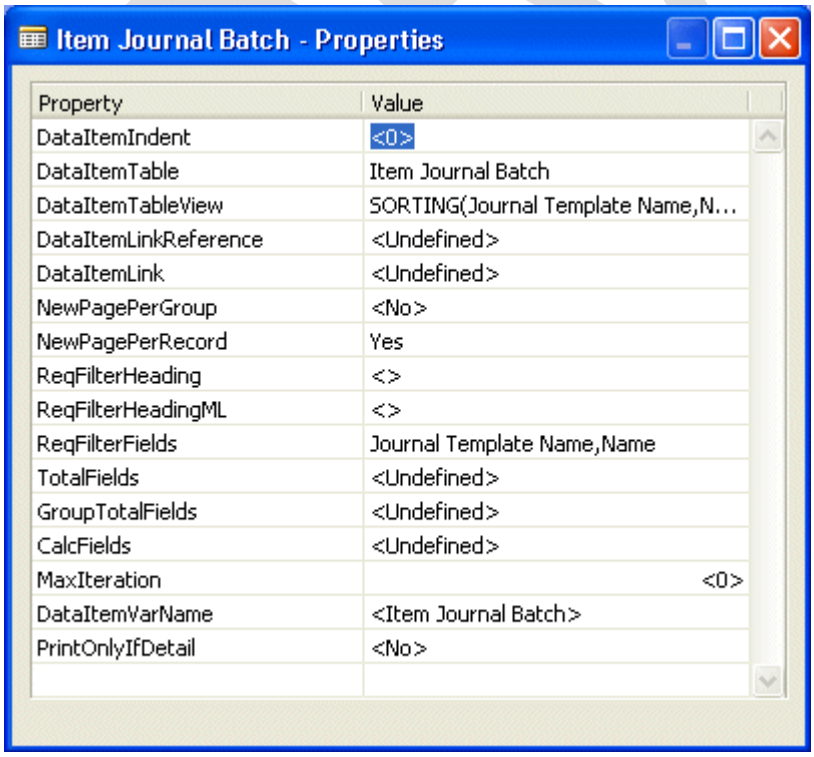
У каждого объекта отчета есть свойства:

- Сам отчет
- Элементы данных (Data items)
- Разделы (Sections)
- Элементы управления в разделах (Controls)
- Форма запроса (Request form)
- Элементы управления на форме запроса (Controls)

Чтобы выбрать объект нужно:

- Чтобы выбрать элемент данных – щелкните по нему в окне ***Report Designer***.
- Чтобы выбрать сам отчет – щелкните по пустой строке в Report Designer или щелкните Правка, Выбрать Объект.
- Чтобы выбрать раздел – в Section Designer щелкните по заголовку раздела или в любом месте раздела (но не по элементам управления)
- Чтобы выбрать элемент управления – щелкните по нему.

Когда объект выбран, щелкните Вид, Properties. Откроется окно ***Properties***:



DRAFT

Каждая строка окна ***Properties*** содержит свойство, значение которого можно установить в поле **Value**. Значение свойства обновляется, когда вы покидаете поле или нажимаете клавишу ENTER. Если значение свойства содержит ошибку (например, вы случайно, установили идентификатор элементу управления такой же, как у другого элемента управления), значение обновлено не будет.

Значения по умолчанию отображаются в угловых скобках < >. Если для свойства есть значение по умолчанию его можно восстановить, для этого надо удалить текущее значение свойства и покинуть поле. Некоторые свойства не имеют значения по умолчанию – обычно это свойства описывающие положение элемента управления на форме. Эти свойства обновляются C/SIDE при перемещении элемента управления.

Наследование Свойств

Элементы управления связанные с полями таблиц наследуют свойства, которые применяются к элементу управления. Например, ваше приложение хранит суммы с точностью в пять знаков после запятой, а в отчет вы хотите выводить данные с двумя знаками после запятой. Для этого нужно изменить свойство *DecimalPlaces* текстовой зоны, вместо унаследованного значения указать свое, которое укажет системе, что нужно выводить данные с меньшим количеством знаков после запятой.

Свойства Отчета

В следующей таблице приведено краткое описание некоторых наиболее важных свойств отчета. Описания всех свойств приведено в онлайновой справке *C/SIDE Reference Guide*. Чтобы получить контекстную справку надо: открыть окно ***Properties,*** поместить курсор на конкретное свойство и нажать F1.

Чтобы открыть окно ***Properties*** для отчета щелкните по пустой строке в Report Designer или щелкните Правка, Выбрать Объект, затем щелкните Вид, Properties (SHIFT+F4).

Свойство	Используется для
ID	Идентификатор отчета. Должен быть уникальным среди отчетов
Name	Наименование отчета
Caption	Заголовок (отображается в форме запроса, значение по умолчанию такое же как Name).
CaptionML	Переводы заголовка
ShowPrintStatus	Должно ли при печати отображаться окно статуса печати (с возможность отменить печать).
UseReqForm	Должна ли форма запроса отображаться перед запуском отчета.
UseSystemPrinter	Если установлено значение Да, для печати отчета система использует системный принтер. Иначе будет использоваться принтер, определенный для комбинации Пользователь/Отчет, которая задается в настройках.
ProcessingOnly	Если установлено значение Да, отчет не будет иметь секций.
TransactionType	Указывается тип транзакции Dynamics NAV. Всего доступно четыре типа базовых транзакций: Browse, Snapshot, UpdateNoLocks и Update. Тип транзакции Report связан с одним из четырех типов транзакции и позволяет отчету использовать доступ к данным в режиме только на чтение, чтобы избежать блокировок. При использовании C/SIDE Database Server, он соответствует Snapshot, а для SQL Server - Browse.

DRAFT

Свойство	Используется для
Description	Описание, используется для внутренних целей. Недоступно конечному пользователю.
TopMargin	Верхнее поле, в 1/100 мм.
BottomMargin	Нижнее поле, в 1/100 мм.
LeftMargin	Левое поле, в 1/100 мм.
RightMargin	Правое поле, в 1/100 мм.
HorzGrid	Расстояние между горизонтальными линиями сетки (1/100 мм).
VertGrid	Расстояние между вертикальными линиями сетки (1/100 мм).
Permissions	Права для отчета на доступ к объектам базы данных. Для отчета могут быть заданы более широкие права, чем для пользователя. Это позволит пользователю напечатать отчет, который основан на данных таблицы, к которой сам пользователь не имеет доступа.
Orientation	Устанавливается ориентация страницы: Портрет (Portrait) или Ландшафт (Landscape).
PaperSize	Задается размер бумаги для данного отчета
PaperSourceFirstPage	Указывается источник бумаги, для печати первой страницы отчета. Полезно если у отчета есть обложка.
PaperSourceOtherPage	Указывается источник бумаги, для печати остальных страниц отчета
DeviceFontName	Данное свойство используется для отчетов, разработанных специально для матричных принтеров, чтобы избежать переключения принтера в графический режим, когда должен печататься текст. В свойстве указывается название шрифта устройства (шрифта, который встроен в принтер).

Свойства Элементов Данных

В следующей таблице приведено краткое описание свойств элементов данных. Описания всех свойств приведено в онлайновой справке *C/SIDE Reference Guide*. Чтобы получить контекстную справку надо: открыть окно ***Properties,*** поместить курсор на конкретное свойство и нажать F1.

Свойство	Используется для
DataItemIndent	Уровень вложенности (можно задать из дизайнера, при создании элемента данных)
DataItemTable	Таблицы, на которой основывается элемент данных (можно задать из дизайнера, при создании элемента данных)
DataItemTableView	Ключ, порядок сортировки, применяемые фильтры
DataItemLinkReference	DataItemVarName элемента данных с меньшим уровнем вложенности. Для подчинения текущего элемента данных другому элементу данных.
DataItemLink	Связь между полями текущего элемента данных и элемента данных указанного в свойстве DataItemLinkReference.
NewPagePerGroup	Должна ли каждая группа печататься на отдельной странице
NewPagePerRecord	Должна ли каждая запись печататься на отдельной странице

DRAFT

Свойство	Используется для
ReqFilterHeading	Заголовок закладки на форме запроса, связанной с текущим элементом данных (по умолчанию выводится название таблицы, на которой основывается элемент данных)
ReqFilterHeadingML	Перевод заголовка закладки на форме запроса.
ReqFilterFields	Названия полей, которые должны быть включены в качестве фильтров на форме запроса
TotalFields	Названия полей, по которым система должна рассчитывать итоги.
GroupTotalFields	Названия полей, на основании которых система будет группировать данные.
CalcFields	Название вычисляемых полей, которые должны быть рассчитаны, после того как запись будет извлечена из базы данных
MaxIteration	Максимальное количество итераций (циклов извлечения записей из базы данных)
DataItemVarName	Название элемента данных в качестве переменной (по умолчанию – название таблицы, на которой основывается элемент данных).
PrintOnlyIfDetail	Печатать элемент данных, только если подчиненный элемент данных содержит данные.

Свойства Разделов

В следующей таблице приведено краткое описание свойств разделов. Описания всех свойств приведено в онлайновой справке *C/SIDE Reference Guide*. Чтобы получить контекстную справку надо: открыть окно ***Properties,*** поместить курсор на конкретное свойство и нажать F1.

Свойство	Используется для
PrintOnEveryPage	Должны ли разделы header и footers печататься на каждой странице
PlaceInBottom	Должен ли раздел footer при печати прижиматься к нижней части странице.
SectionWidth	Ширина в 1/100 мм
SectionHeight	Высота в 1/100 мм

Свойства Элементов Управления

У элементов управления в отчетах такие же свойства, как у элементов управления в формах. Описания всех свойств приведено в онлайновой справке *C/SIDE Reference Guide*. Чтобы получить контекстную справку надо: открыть окно ***Properties,*** поместить курсор на конкретное свойство и нажать F1.

13.2 Создание Простого Отчета

В данном примере будет показано, как создать очень простой отчет, который выводит список клиентов. Отчет основывается на одной таблице, которая содержит всю необходимую информацию.

Определение Модели Данных

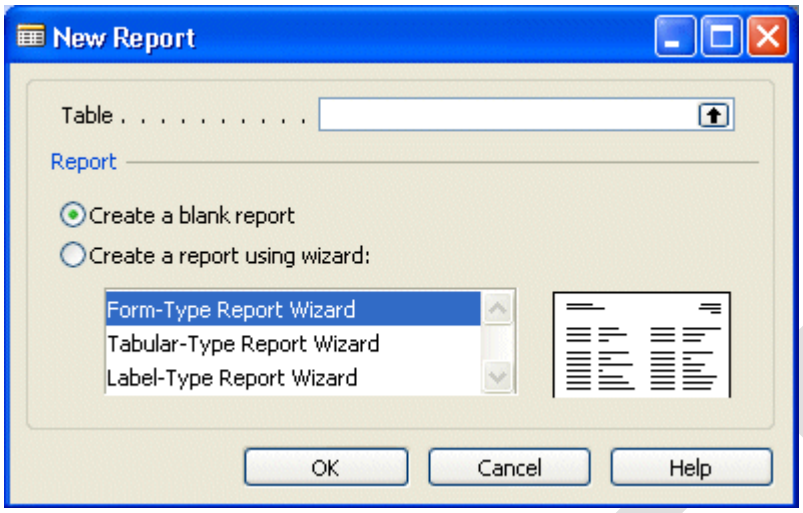
Первым шагом при создании отчета является определение модели данных, на которой отчет будет основываться. Определение модели данных осуществляется созданием элементов данных.

DRAFT

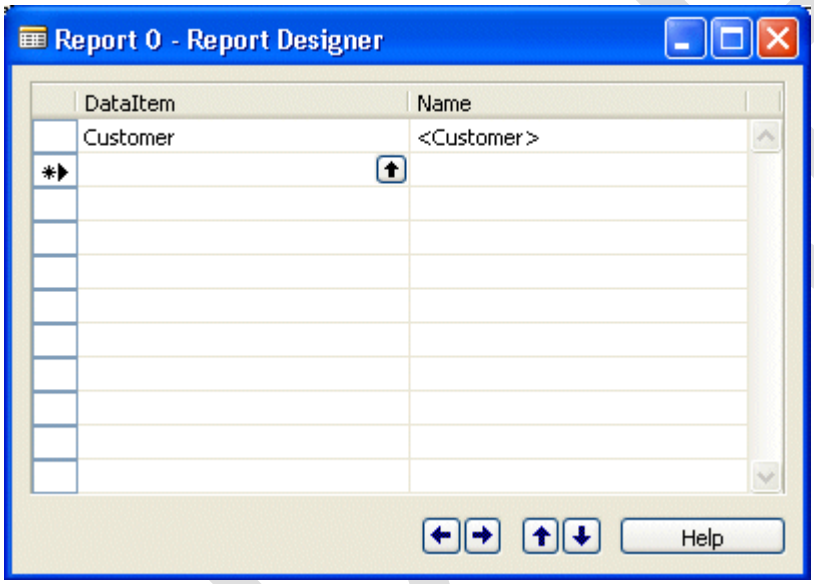
Чтобы создать элемент данных:

1. Щелкните Сервис, Object Designer (SHIFT+F12).

2. В окне **Object Designer** щелкнуть кнопку **Report**, затем кнопку **New**. Откроется окно ***New Report***:

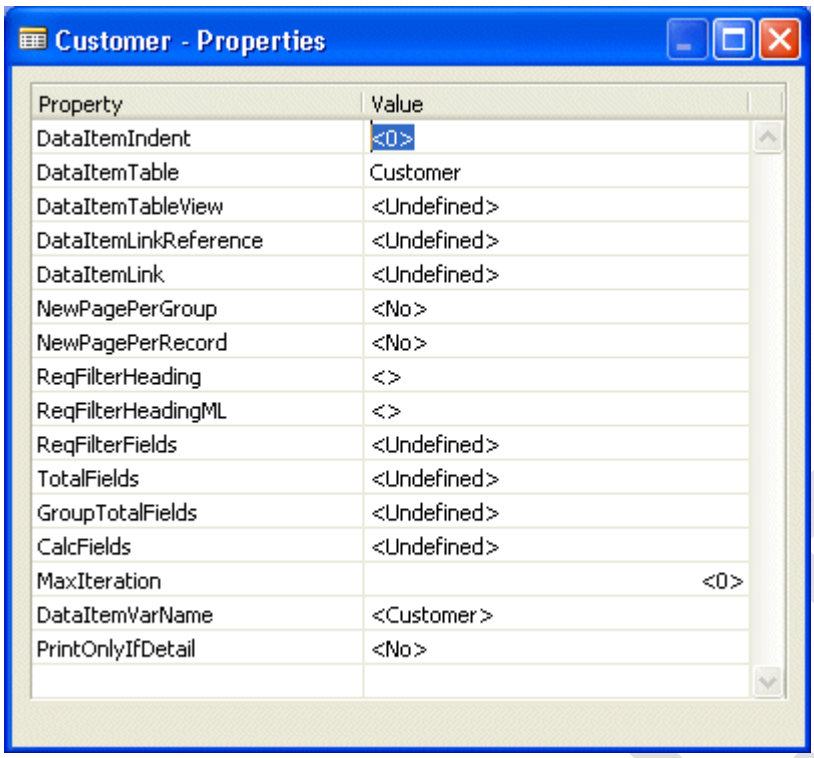


3. В разделе **Report** выберите опцию **Create a blank report**, затем щелкните кнопку OK. Откроется окно ***Report Designer***:

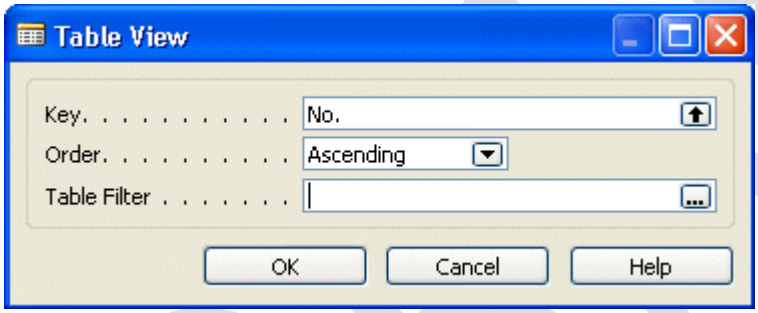


4. Щелкните кнопку Выбор в поле **Data Item** первой строки, чтобы выбрать таблицу из списка ***Table List***. В данном примере, выберите таблицу ***Customer (18)***.

5. Щелкните Вид, Properties (SHIFT+F4) чтобы открыть окно ***Properties*** для данного элемента данных.

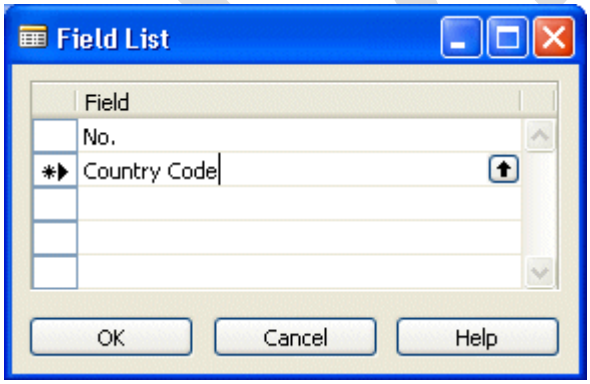


6. Щелкните кнопку AssistButton в свойстве *DataItemTableView*, чтобы открыть окно ***Table View***:



7. Укажите ключ, порядок сортировки и требуемые фильтры, затем щелкните кнопку OK. В данном примере, в качестве ключа выбрано поле **No.**, установлена сортировка по возрастанию (Ascending). Поле **Table Filter** оставлено пустым, т.е. для данной таблицы постоянный фильтр не установлен.

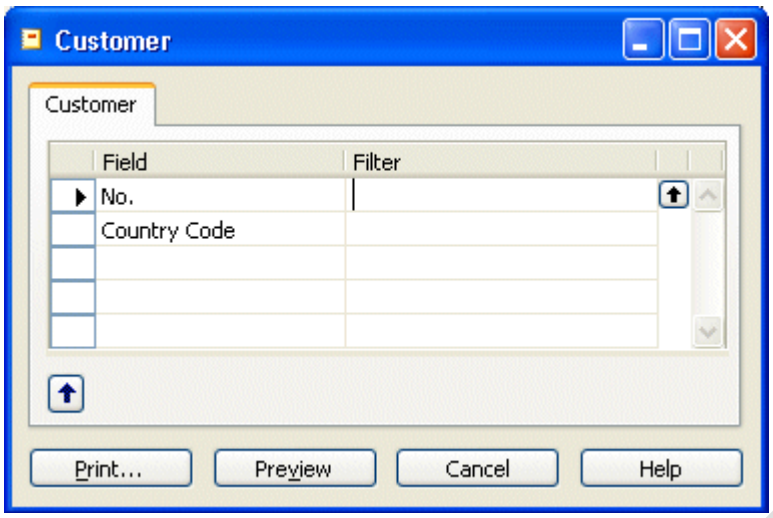
8. Щелкните кнопку AssistButton в свойстве *ReqFilterFields,* чтобы открыть окно ***Field List***:



9. Выберите поля, по которым чаще всего пользователям требуется устанавливать фильтры. Для выбора можно использовать кнопку Выбор, в правой части поля. В данном примере выбраны поля **No.** и **Country/Region Code**. Щелкните кнопку OK.

10. Щелкните Файл, Save, чтобы сохранить отчет. Затем запустите отчет из **Object Designer**, чтобы посмотреть форму запроса данного отчета. Как вариант можно щелкнуть Файл, Run.

На следующем рисунке показано, как должна выглядеть форма запроса:



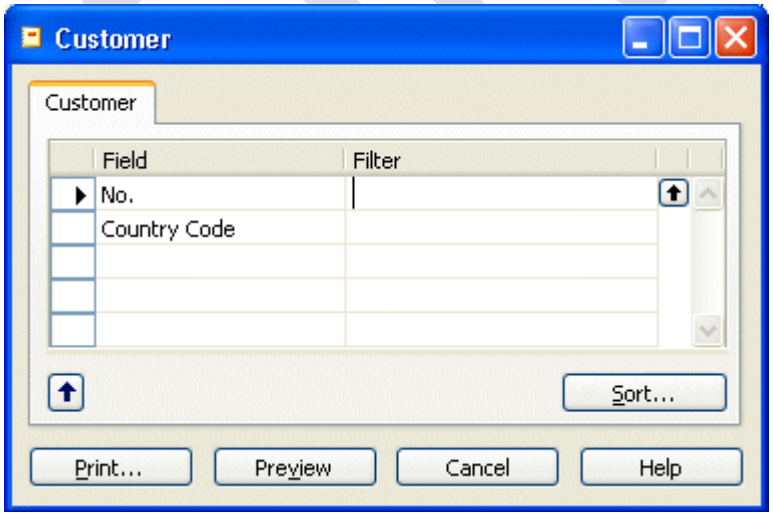
Так как ключ и порядок сортировки были определены при создании отчета, то пользователь может только ввести фильтры.

На форме показаны поля, которые были определены в свойстве ReqFilterFields, однако пользователь сможет добавить нужные ему фильтры самостоятельно (добавив строки в форме). Тем не менее, указание на форме часто используемых для фильтрации полей является хорошей идеей. При проектировании отчета, следует стараться соблюсти баланс. Например, если в таблице много полей, неопытный пользователь может испытать затруднения при поиске подходящего поля, на которое он сможет наложить фильтр.

Закладку, где устанавливаются фильтры можно не показывать вовсе. Для этого надо оставить свойство ReqFilterFields пустым, а в свойстве DataItemTableView указать ключ и порядок сортировки.

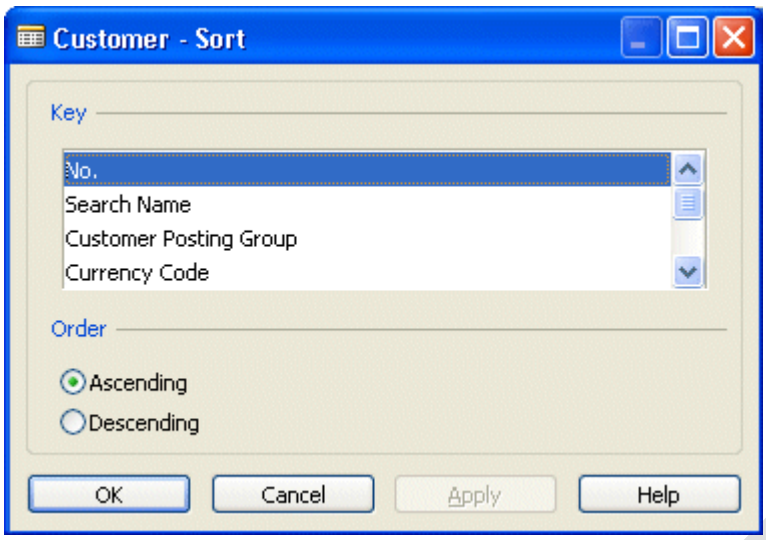
Если вы создали форму запроса, то она будет показана, иначе будет показана пустая форма, на которой пользователи смогут выбрать Печать, Отмена и т.д. Если для отчета установлено свойство UseReqForm = Нет, при запуске отчета сразу начнется печать. В этом случае пользователь не сможет передумать и отказаться от запуска отчета (тем не менее, он может отменить печать, хотя при этом несколько страниц могут напечататься).

Если свойство DataItemTableView не определено, пользователи также могут установить ключ и порядок сортировки. В этом случае форма запроса будет выглядеть следующим образом:



Если щелкнуть кнопку **Сортировка**, то можно будет выбрать ключ и порядок сортировки в следующем окне:

DRAFT



Примечание

Будьте внимательны при предоставлении пользователям возможности изменять ключи и сортировку. В сложных отчетах, которые основываются на нескольких таблицах, функциональность может зависеть от определенного ключа и порядка сортировки. Кроме того, предоставление пользователям возможности накладывать произвольные фильтры может не соответствовать логике отчета.

В простых отчетах, таких как в текущем примере, выбор ключа и порядка сортировки можно предоставить пользователю.

Использование Мастера

В окне ***New Report*** в разделе **Report** можно выбрать одного из мастеров, например, Form-Type Report Wizard.

Мастер поможет выбрать поля, на которых основывается отчет, а также порядок сортировки. Обратите внимание, в содержимое поля **Available Fields** на первой странице мастера отображает значения свойства *Caption*, а не *Name*.

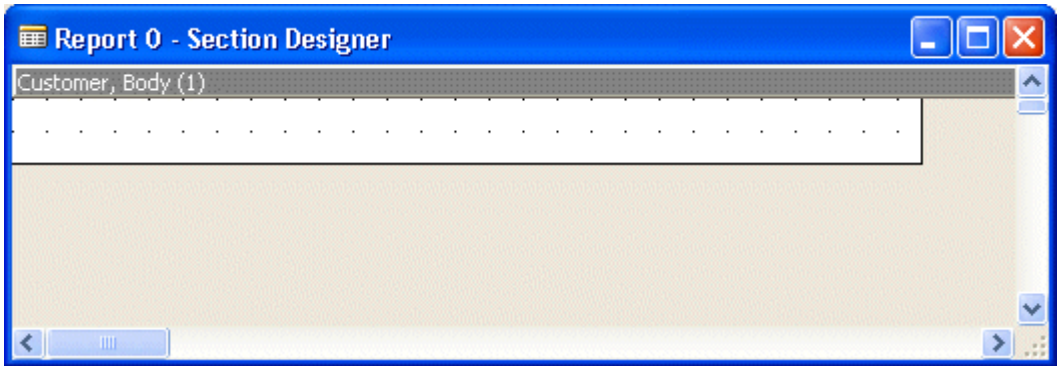
Создание Разделов

После того как в отчете была определена модель данных, можно перейти к следующему шагу – созданию разделов.

Чтобы создать раздел:

1. Пока активно окно ***Report Designer*** нужно щелкнуть Вид, Sections. Откроется окно **Section Designer**.

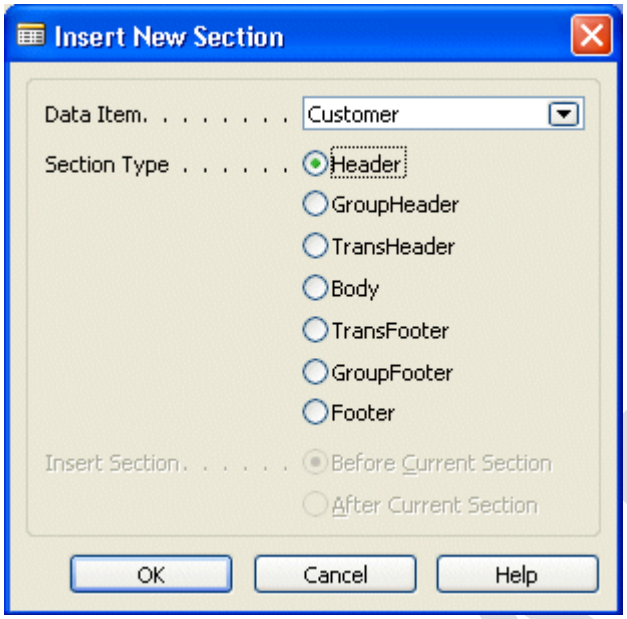
После того как был создан элемент данных (как было описано в предыдущем разделе), **Section Designer** будет выглядеть следующим образом:



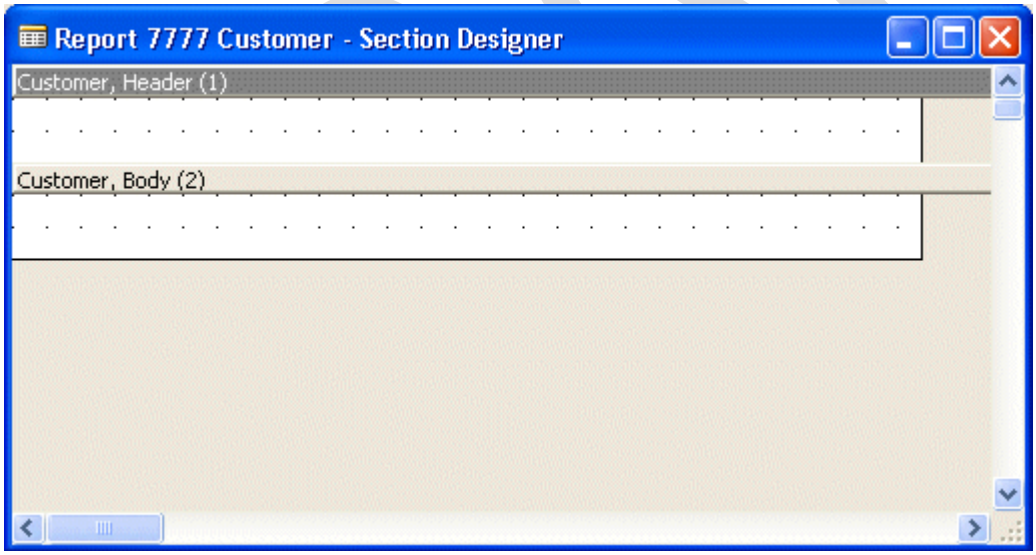
DRAFT

2. Как можно видеть уже присутствует раздел "**Customer,Body (1)**" ("1" означает, что это первый раздел элемента данных). По умолчанию раздел Body вставляется автоматически при создании элемента данных. Эти разделы появляются в таком же порядке, как элементы данных в Report Designer.

3. Щелкните Правка, Создать, чтобы добавить раздел header для элемента данных Customer. Появится окно ***Insert New Section***:

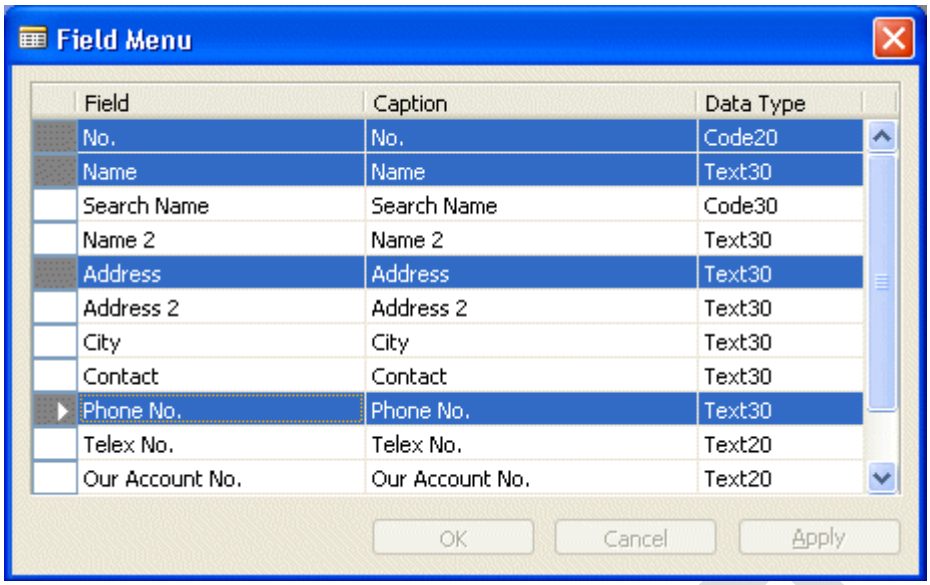


4. Выберите Header в качестве типа раздела и щелкните кнопку OK. Теперь Section Designer будет выглядеть следующим образом:

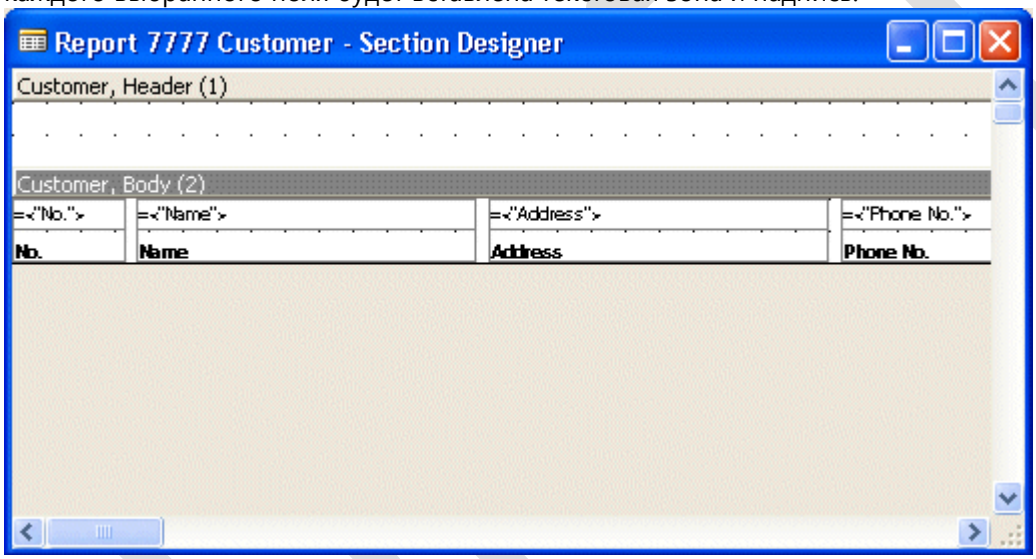


5. Щелкните Вид, Field Menu, чтобы открыть инструмент Field Menu. Выберите поля, которые должны присутствовать в отчете (можно выбрать несколько полей, для этого при выборе нужно удерживать нажатой клавишу CTRL).

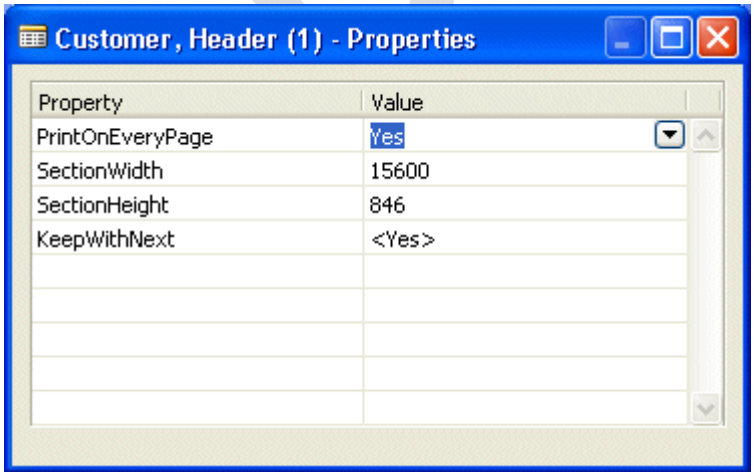
В следующем примере, выбрано четыре поля:



6. Переместите курсор мыши в раздел Body. Щелкните одни раз, чтобы активизировать окно, курсор изменит вид на «вставка». Поместите курсор в левую часть раздела и щелкните еще раз. Для каждого выбранного поля будет вставлена текстовая зона и надпись.



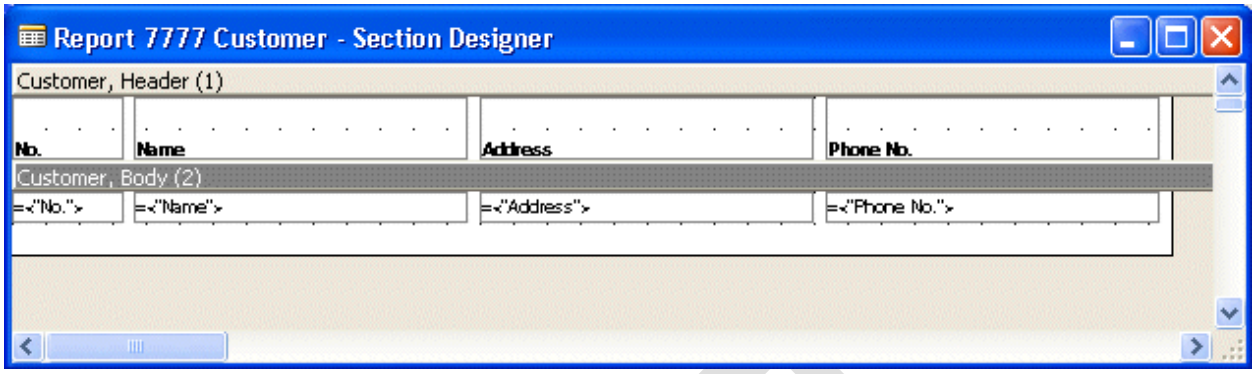
7. Щелкните Вид, Properties (SHIFT+F4) чтобы открыть окно ***Properties***. Щелкните по разделу Header, чтобы убедиться, что окно ***Properties*** открыть для него. Посмотрите значение свойства SectionWidth (в 1/100 мм).



Ширина всех секций была изменена таким образом, чтобы включать все вставленные элементы управления (ширина по умолчанию, пока не было вставлено ни одного элемента управления,

составляет 12000). В данном примере ширина составила 15450, или 15.45 см. Если отчет планируется печатать на бумаге формата А4, то этот размер подходит, т.к. ширина бумаги формата А4 составляет 21 см.

8. Выберите все надписи (label) в разделе Body (при выборе удерживайте нажатой клавишу CTRL), а затем переместите их все в раздел Header.

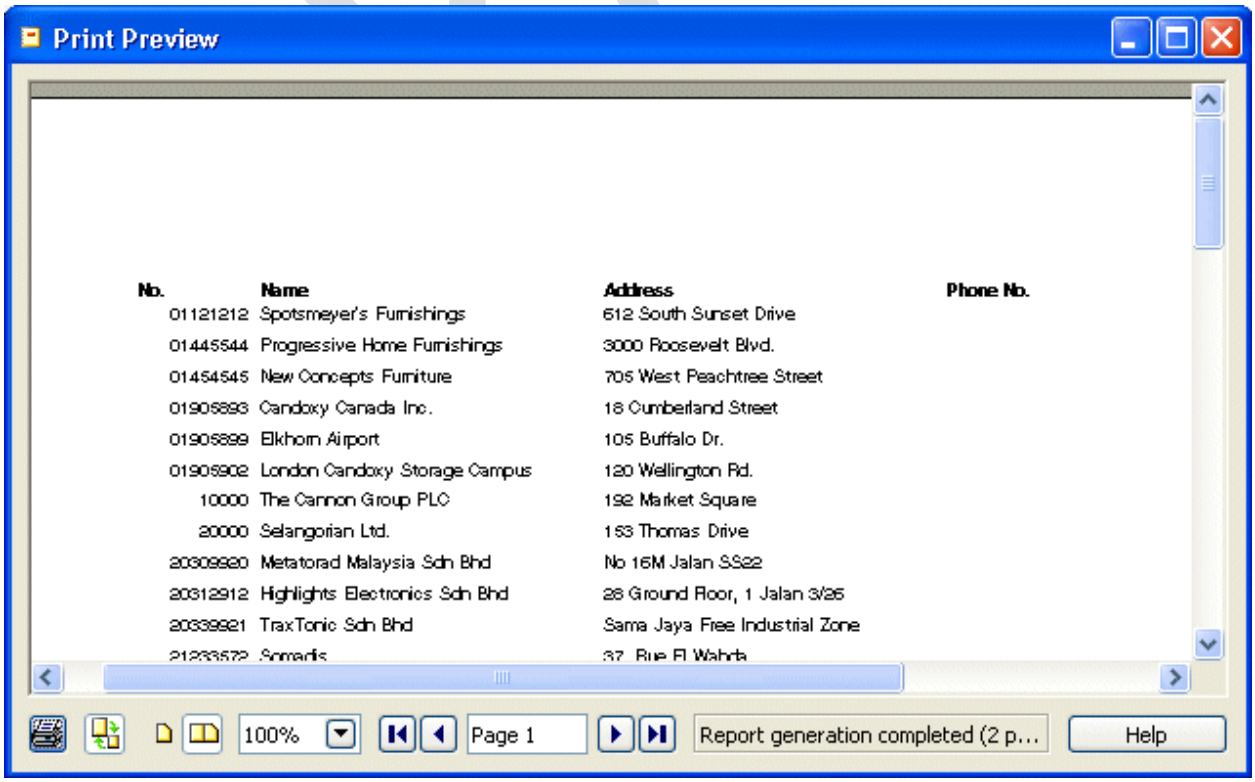


Теперь отчет, может быть напечатан, однако чтобы при печати он выглядел хорошо, нужно выполнить еще ряд операций.

9. Если отчет будет состоять из нескольких страниц, будет хорошо, если раздел Header, в котором содержаться надписи, будет печататься на каждой странице. Откройте окно ***Properties*** для раздела Header и установите в свойстве *PrintOnEveryPage* значение Да.

10. После того как надписи были перенесены из раздела Body, она стала слишком высокой, в результате при выводе на печать каждого клиента, появляются пустые строки. Чтобы изменить размер раздела Body, передвигайте курсор мыши, пока он не коснется нижней границы раздела Body и не изменит свой вид на «изменить размер по вертикали». Затем щелкните и перетащите границу вверх, пока высота раздела не будет такой же, как высота текстовых зон.

11. Закройте и сохраните отчет, затем запустите его из **Object Designer**. Щелкните кнопку **Просмотр**, чтобы посмотреть на результат:



Как можно видеть, что выравнивание данных в столбце **No.** и надписи не совпадают. Причина заключается в том, что для обоих элементов управления используется выравнивание General (по

умолчанию). Надпись прижата влево, так как содержит текст, а данные в текстовых зонах вправо, так как они содержат числа.

Простое решение, заключается в выравнивании надписей по правому краю.

Чтобы изменить выравнивание для элемента управления надпись:

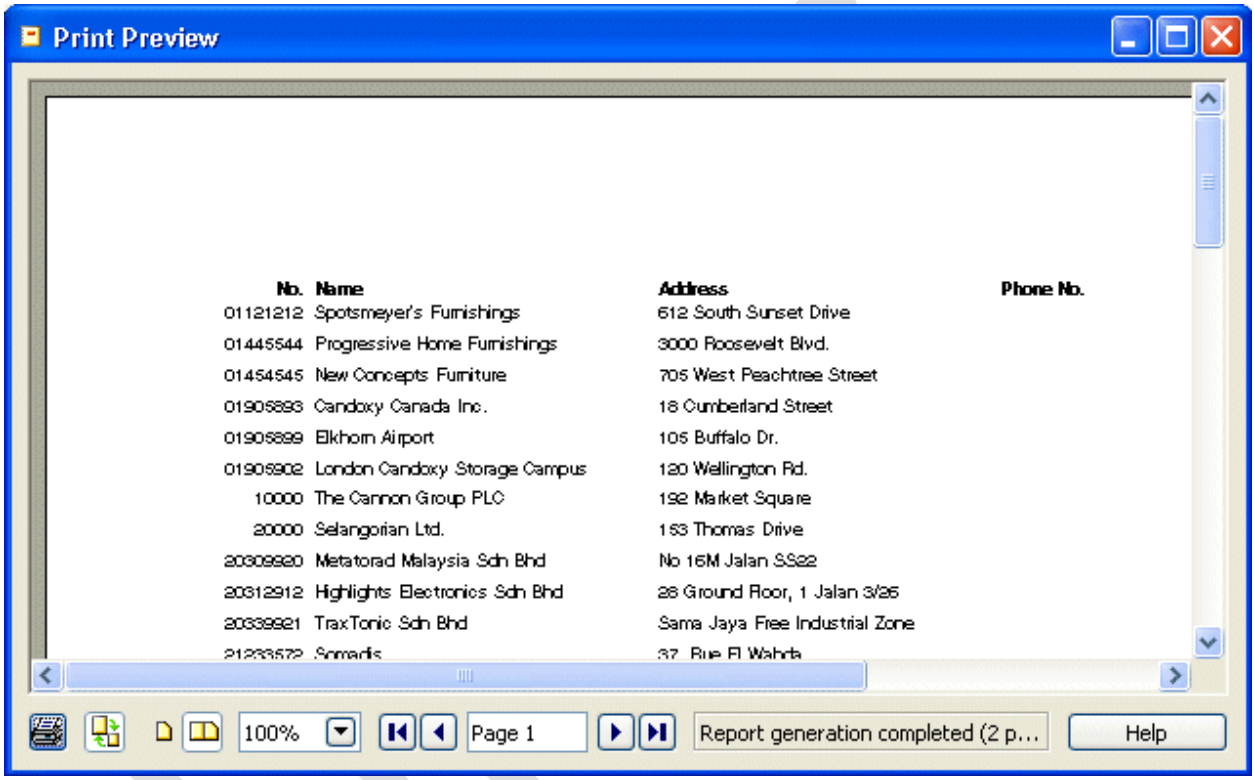
1. Открыть отчет в **Report Designer**, открыть **Section Designer**.

2. Выбрать надпись No. и открыть для нее окно ***Properties***. В свойстве *HorzAlign* установить значение Right.

3. Закрыть и сохранить отчет.

4. Запустить отчет из Object Designer и щелкнуть кнопку Просмотр.

Теперь отчет выглядит следующим образом:



13.3 Создание Более Сложного Отчета

Отчет, созданный в предыдущем примере, был очень простым: он основывался всего на одной таблице. В данном разделе вы узнаете, как создавать отчеты, которые основаны на нескольких таблицах.

Отчет, который будет создан, в данном примере использует две таблицы: первая таблица ***Customer***, которую использовали в предыдущем примере, вторая таблица - ***Sales Line***, в ней содержатся неучтенные строки заказов продажи (информация о товарах заказанных клиентами). Между данными таблицами существует связь «один-ко-многим»: для одного клиента может существовать много строк заказа продажи, а для строки заказа продажи может быть указан только один клиент.

Определение Модели Данных

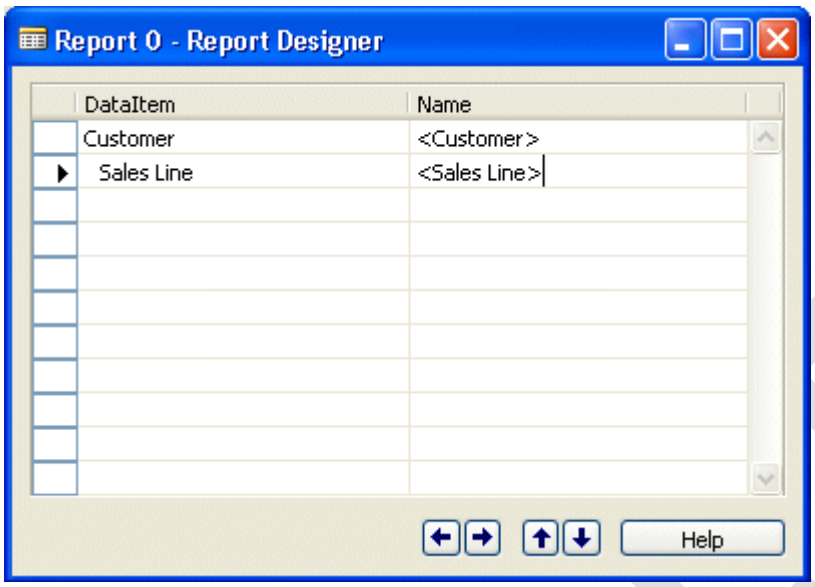
Техническое описание шагов предпринимаемых при создании отчета было приведено ранее.

Чтобы определить модель данных:

1. Откройте **Object Designer** (SHIFT+F12), щелкните кнопку **Report**, затем кнопку **New**. Создайте пустой отчет.

2. В окне ***Report Designer*** укажите для первого элемента данных таблицу ***Customer***, а для второго таблицу ***Sales Line***.

3. Выберите элемента данных ***Sales Line*** и щелкните один раз кнопку со стрелкой вправо, чтобы задать вложенность:



Такая модель данных работает следующим образом:

- Отчет обрабатывает записи таблицы **Customer**.
- Для каждой записи в таблице **Customer** отчет обрабатывает весь набор записей в таблице **Sales Line**.

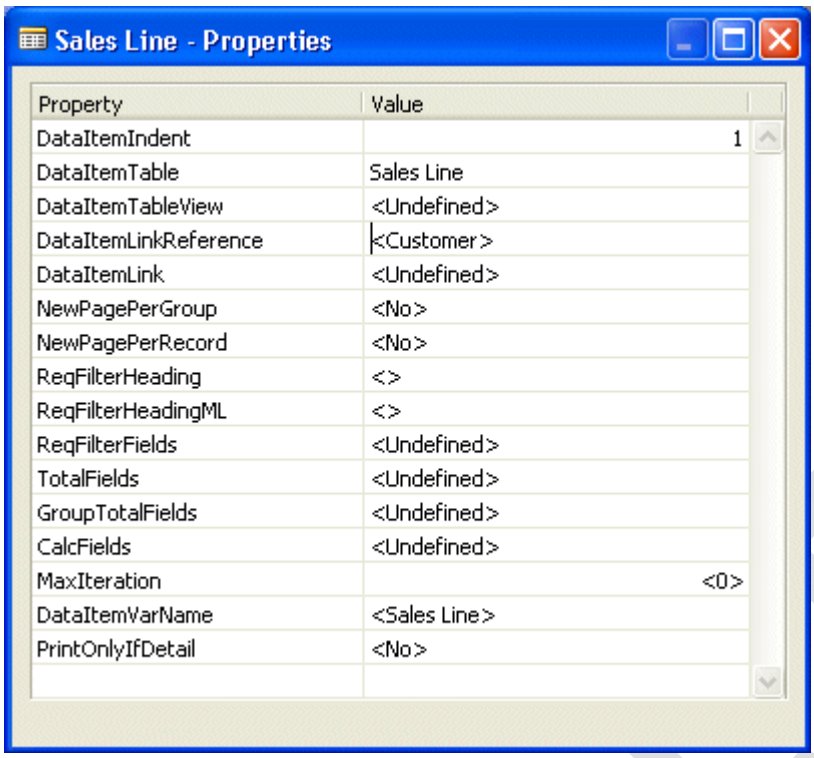
Это не совсем то, что требуется от данного отчета. В отчет должны попадать только те записи таблицы **Sales Line**, которые связаны с текущим клиентом (записью таблицы **Customer**). Это задается с помощью свойств *DataItemLink* и *DataItemLinkReference*.

Свойство *DataItemLinkReference* указывает на элемент данных более высокого уровня (с меньшим значением вложенности), а в свойстве *DataItemLink* определяются поля, на основании которых выполняется связь между двумя элементами данных. В данном примере, должны отображаться только те записи таблицы ***Sales Line,*** у которых значение в поле Sell-to Customer No. равно значению No. таблицы ***Customer***.

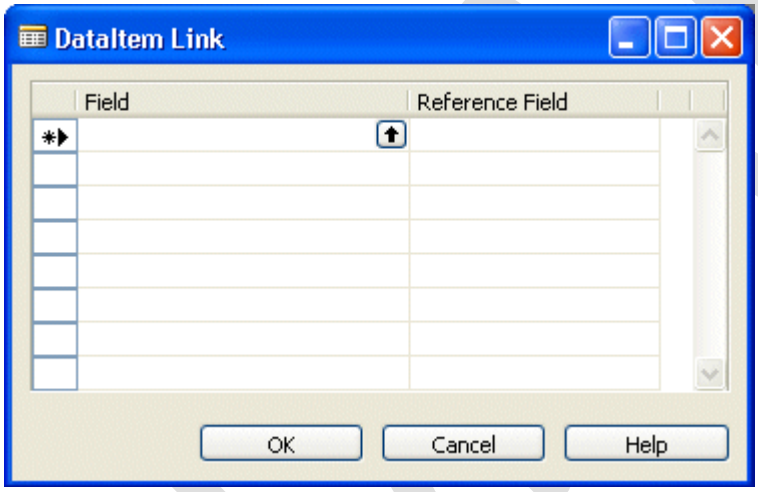
4. Откройте окно ***Properties*** для элемента данных **Sales Line**.

5. В свойстве *DataItemLinkReference* укажите элемент данных с меньшим уровнем вложенности (**Customer**), с которым должен быть связан элемент данных с большим уровнем вложенности (**Sales Line**). В большинстве случаев данное свойство заполняется системой автоматически.

DRAFT

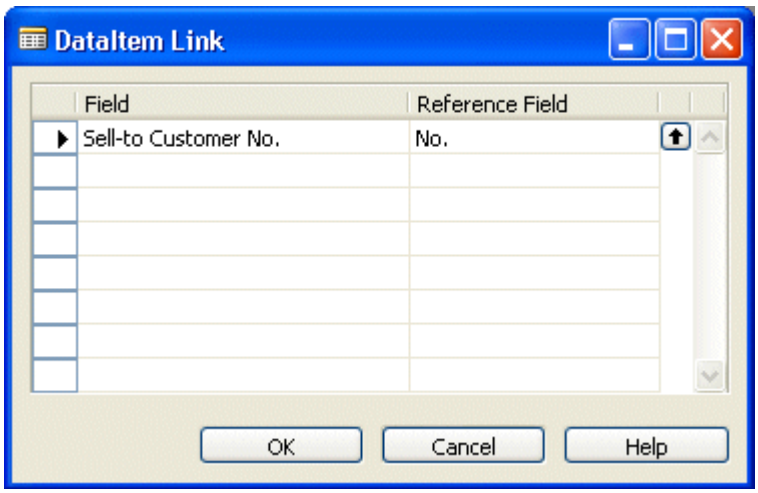


6. Щелкните кнопку AssistButton в свойстве *DataItemLink*. Откроется окно ***DataItem Link***:



7. В поле **Field** введите наименование поля из таблицы ***Sales Line*** (вложенный элемент данных), которое должно соответствовать полю таблицы ***Customer***. В данном примере, укажите поле **Sell-to Customer No.**

8. В поле **Reference Field** введите наименование поля таблицы ***Customer***, которому должно соответствовать поле из таблицы ***Sales Line***. В данном примере, укажите поле **No**. Щелкните кнопку OK.



9. Теперь, откройте окно ***Properties*** для элемента данных Customer и установите в свойство *PrintOnlyIfDetail* значение Да. Это означает, что раздел Body для элемента данных Customer будет отображаться только в том случае, если в таблице ***Sales Line*** есть соответствующие записи.

Теперь модель данных будет выглядеть следующим образом:

- Отчет обрабатывает записи таблицы Customer.
- Для каждой записи в таблице Customer отчет выбирает записи из таблицы **Sales Line** при условии, что поле в **Sell-to Customer No.** такое же значение, как и в поле **No.** таблицы **Customer**.
- Если для таблицы **Customer** нет соответствующих записей в таблице **Sales Line**, никакая информация по клиенту не будет напечатана.

Создание Разделов

Так как ранее уже было приведено описание того, как следует создавать разделы для отчетов с одним элементом данных, то далее акцент будет на создании разделов для вложенных элементов данных.

Чтобы создать разделы:

1. При открытии Section Designer в нем уже будут созданы разделы Body для каждого элемента данных. Щелкните Правка, Создать и вставьте новый раздел Header для элемента данных Customer.

2. Щелкните Вид, Field Menu, чтобы открыть окно ***Field Menu*** и добавить поля **No.**, **Name**, **Address** и **Phone No.** в раздел **Customer, Body (2)**. Переместите надписи в раздел Header.

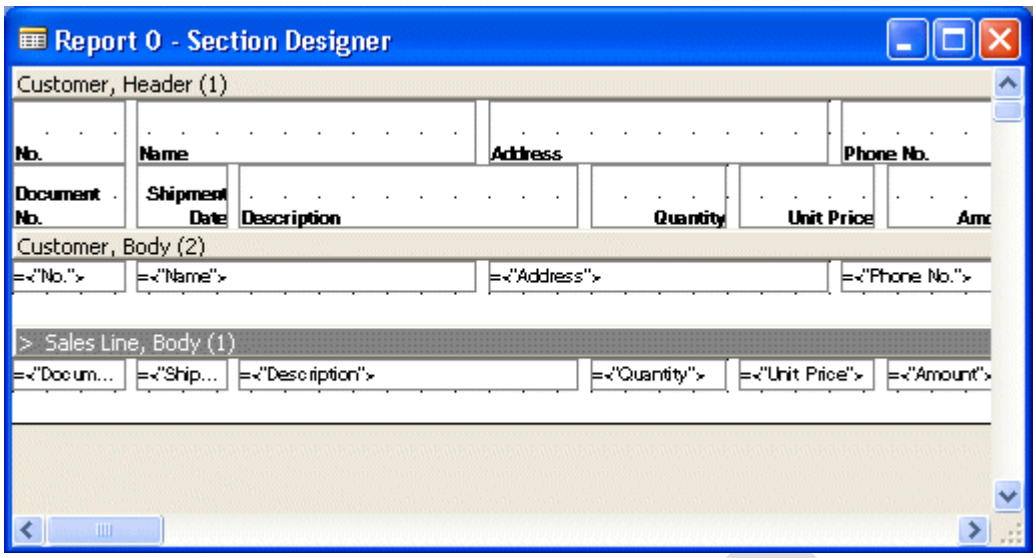
Процедура полностью аналогична той, которая использовалась для создания простого отчета.

3. Откройте окно ***Field Menu*** и добавьте поля **Document No.**, **Shipment Date**, **Description**, **Quantity**, **Unit Price** и **Amount** в раздел **Sales Line, Body (1)**.

4. Сейчас нужно решить, что делать с надписями для элементов управления в разделе **Sales Line, Body (1)**. Если их оставить в данном разделе, то они будут печататься для каждой записи. Если для элемента данных Sales Line добавить раздел header, то он будет печататься, каждый раз при открытии выборки. Выборка открывается для каждой записи элемента данных Customer.

Оба решения не очень хороши, поэтому можно применить третий подход и переместить надписи в раздел header section элемента данных Customer:

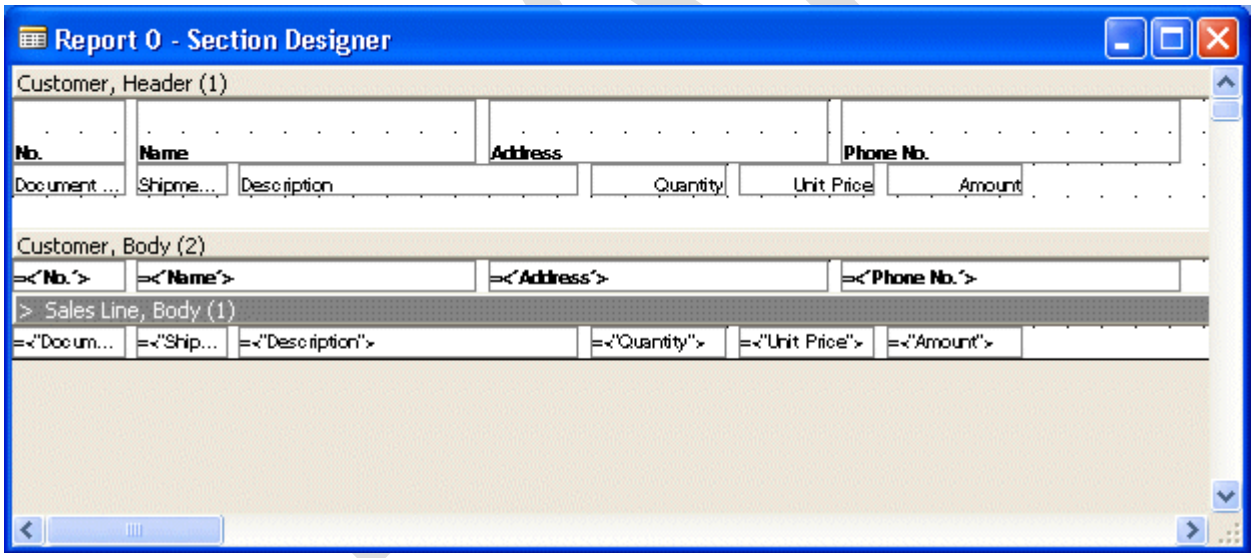
DRAFT



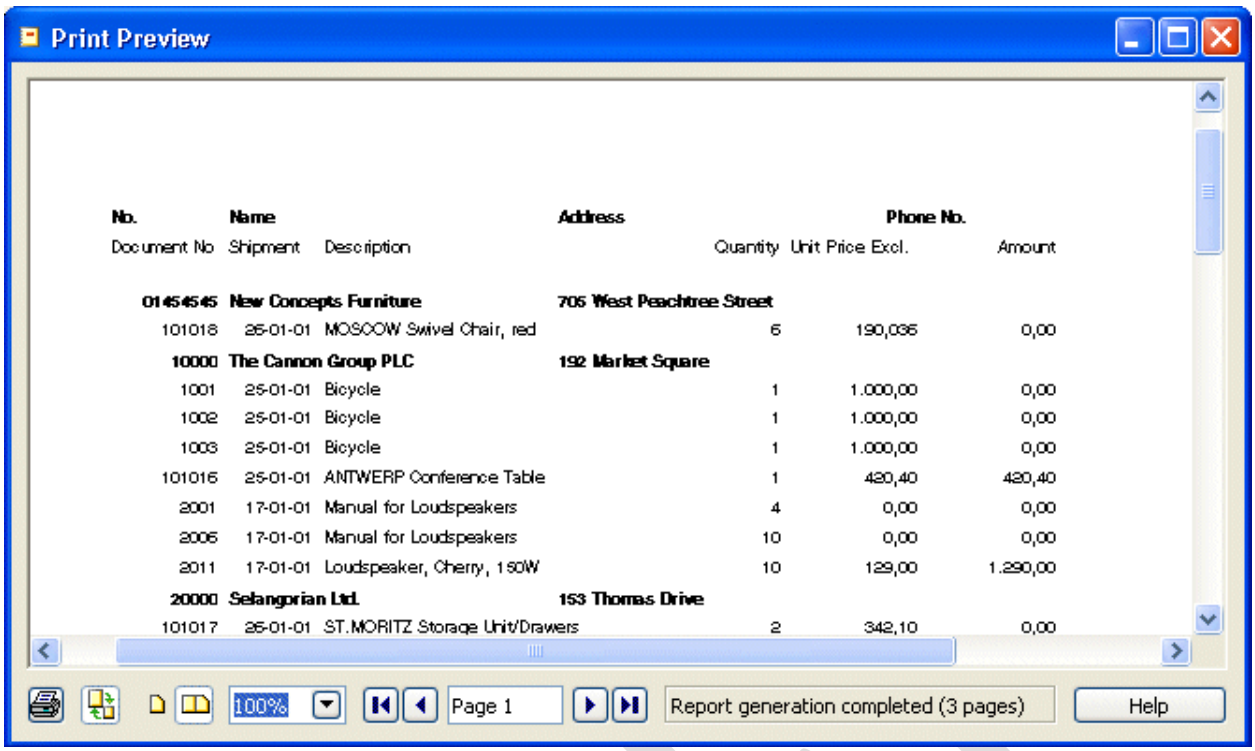
5. Установите свойство *PrintOnEveryPage* = *Да* раздела **Customer, Header (1)**.

Теперь надписи и для записей Customer и для записей Sales Line будут печататься в качестве заголовков столбцов раздела **Customer, Header (1)**.

Чтобы сделать связь между надписями и данными более прозрачной, изменим шрифт для надписей, использующихся в качестве заголовков столбцов для записей из таблицы Sales Line. Вместо жирного шрифта, используемого по умолчанию, установите обычный шрифт. Для текстовых зон элемента данных Customer установите жирный шрифт, чтобы выделять данные записи среди напечатанных строк (записей из таблицы **Sales Line** будет больше, чем из таблицы **Customer**). Затем измените размер надписей для таблицы **Sales Line** так, чтобы они занимали только одну строку.



6. Сохраните и закройте отчет, затем запустите отчет из **Object Designer**. Отчет должен выглядеть следующим образом:



Print Preview

No.	Name		Address		Phone No.	
Document No	Shipment	Description		Quantity	Unit Price Excl.	Amount
01454545	**New Concepts Furniture**		**705 West Peachtree Street**			
101018	25-01-01	MOSCOW Swivel Chair, red		6	190,035	0,00
10000	**The Cannon Group PLC**		**192 Market Square**			
1001	25-01-01	Bicycle		1	1.000,00	0,00
1002	25-01-01	Bicycle		1	1.000,00	0,00
1003	25-01-01	Bicycle		1	1.000,00	0,00
101016	25-01-01	ANTWERP Conference Table		1	420,40	420,40
2001	17-01-01	Manual for Loudspeakers		4	0,00	0,00
2006	17-01-01	Manual for Loudspeakers		10	0,00	0,00
2011	17-01-01	Loudspeaker, Cherry, 150W		10	129,00	1.290,00
20000	**Selangorian Ltd.**		**153 Thomas Drive**			
101017	25-01-01	ST.MORITZ Storage Unit/Drawers		2	342,10	0,00

100% Page 1 Report generation completed (3 pages) Help

DRAFT

ГЛАВА 14 РАСШИРЕНИЕ ФУНКЦИОНАЛЬНОСТИ ОТЧЕТОВ

В данной главе будет рассказано, как группировать данные и создавать итоги. Также будет приведено описание триггеров отчетов и полезных возможностей Report Designer.

- Создание группировок и итогов
- Триггеры в отчетах
- Создание отчета с использованием виртуальной таблицы
- Создание простого документа
- Создание непечатаемого отчета
- Типы отчетов

14.1 Создание Группировок и Итогов

При создании полезных отчетов важен способ, которым вы группируете данные и рассчитываете по ним итоги. Отчет, который содержит итоги и группировки, предоставляет пользователю информацию, которую он не может получить непосредственно из системы.

Второй отчет, который был создан в предыдущей главе, основывался на двух таблицах и включал информацию о списке клиентов и соответствующих им записях из таблицы ***Sales Line***. У вас есть несколько способов, чтобы увеличить ценность данной информации.

Во-первых, если задача отчета предоставлять статистику, то будет полезнее сгруппировать строки продажи по товарам (группировка по номеру товара), вместо того, чтобы печатать все строки из всех документов продажи. Каждая строка должна содержать итоговое количество и общую сумму по каждому товару, заказанному клиентом.

Во-вторых, будет полезно отобразить итог по каждому клиенту, где показать общую сумму заказа.

Этот отчет будет создан в данной главе.

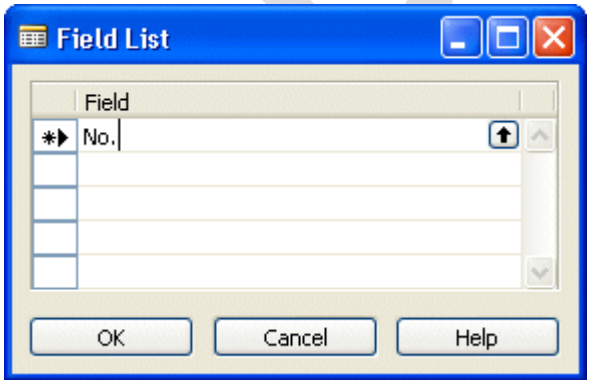
Определение Модели Данных

Продолжим работать с отчетом, созданным в предыдущей главе.

Чтобы добавить итоги и группировки нужно выполнить следующие шаги:

1. Открыть окно ***Properties*** (SHIFT+F4) для элемента данных Sales Line.

2. В свойстве *GroupTotalFields* ввести название поля, по которому будут группироваться записи. В данном примере используется поле **No.**:

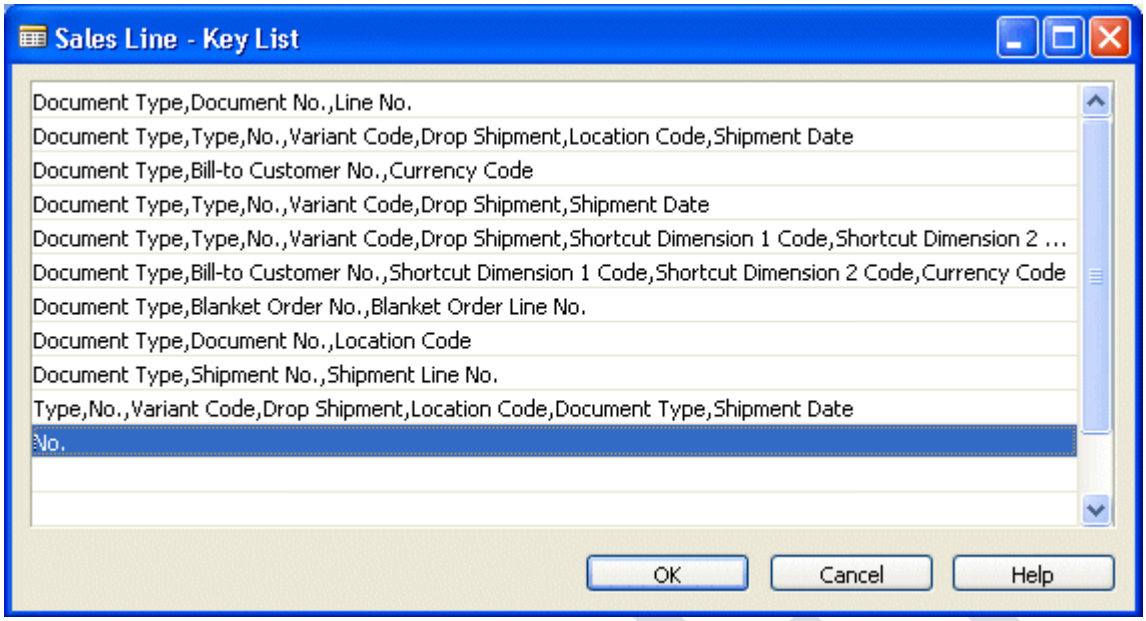


3. Щелкните кнопку AssistButton в свойстве *DataItemTableView,* чтобы выбрать ключ. Нужно выбирать ключ, содержащий поля, на основании которых будет выполняться группировка.

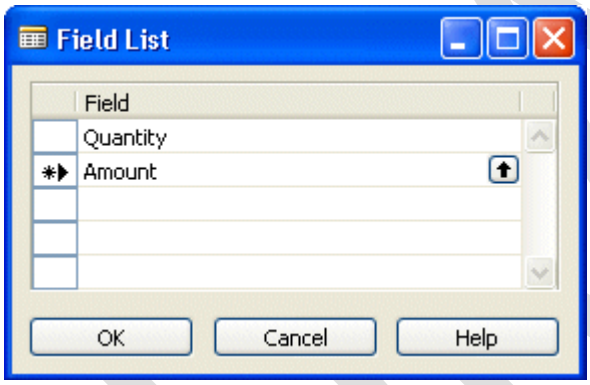
Если вы выбрали ключ, состоящий из нескольких полей, группировка может не работать. Это произойдет если перед полем, на основании которого будет выполняться группировка, есть другие

поля и значения в данных полях различаются. Возможно, для такого отчета придется создать отдельный ключ.

В данном отчете используется ключ, состоящий из поля **No.** (создан для таблицы ***Sales Line)***:

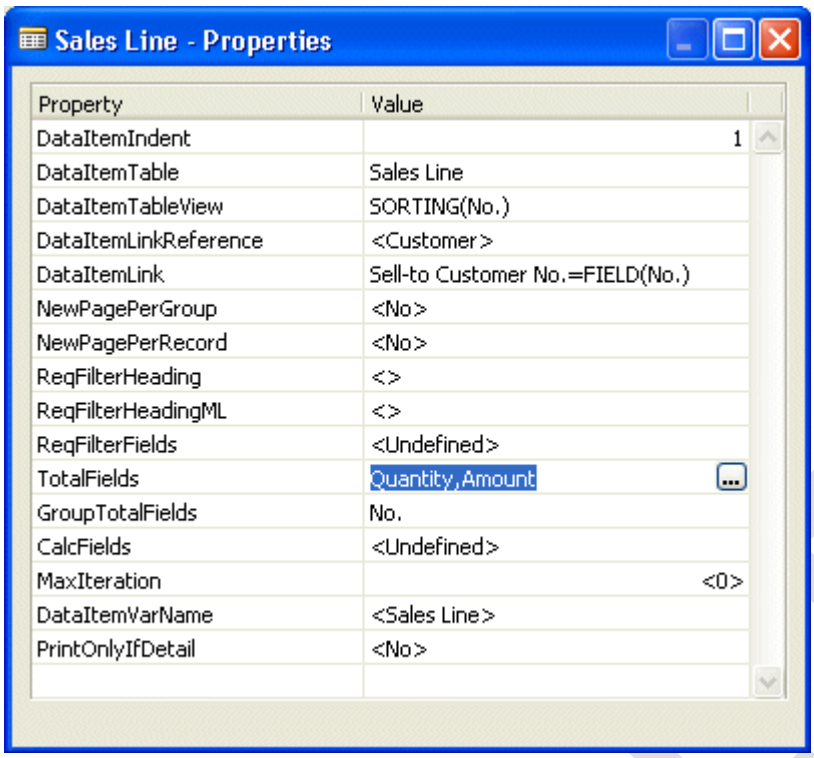


4. Щелкните кнопку AssistButton в свойстве *TotalFields,* чтобы выбрать поля, по которым должны рассчитываться итоги. В данном примере это поля **Quantity** и **Amount**:



Теперь окно ***Properties*** для элемента данных Sales Line должно выглядеть следующим образом:

DRAFT



Модель данных определена:

- Для каждой записи таблицы Customer выбираются записи таблицы Sales Line, связанной с выбранным клиентом.
- Записи в элементе данных Sales Line группируются согласно номером товаров.
- Для элемента данных Sales Line поддерживаются итоги по полям **Quantity** и **Amount**.

Связь между Итогами и Разделами

В данном отчете была установлена иерархия, согласно которой элемент данных Customer относится к более высокому уровню, а элемент данных Sales Line был вложенным. Далее записи таблицы Sales Line группируются по полю **No.**, а по полям **Amount** и **Quantity** рассчитываются итоги.

Каковы взаимосвязи между данными итогами и разделами, как данные итоги будут напечатаны?

До этого момента использовались только разделы Header и Body. Чтобы печатать итоги, следует использовать другие типы разделов:

Название Раздела	Выход
Header	Заголовок. Этот раздел печатается первым для элемента данных. По умолчанию он печатается только один раз при каждой обработке элемента данных, но вы можете выбрать возможность печатать его на каждой странице (свойство *PrintOnEveryPage* = Да).
Body	Область данных. Этот раздел печатается один раз для каждой записи, которая обрабатывается элементом данных. Если запись пропущена или не попала в фильтр, этот раздел не печатается для данной записи.

DRAFT

Название Раздела	Выход
Footer	Нижний колонтитул. Этот раздел печатается последним для элемента данных. По умолчанию он печатается только один раз при каждой обработке элемента данных, но вы можете выбрать возможность печатать его на каждой странице (*PrintOnEveryPage* = Да). Вы можете также определить, чтобы этот раздел печатался в нижней части страницы, что гарантирует, что раздел будет помещен как можно ближе к нижней границе страницы (*PlaceInBottom* = Да)
GroupHeader	Групповой заголовок. Этот раздел печатается прямо перед первой записью в группе, которая подвергается обработке. Этот раздел будет печататься ТОЛЬКО в случае, если свойство GroupTotalFields для элемента данных установлено, а элемент данных соответственно отсортирован.
GroupFooter	Групповой нижний колонтитул. Этот раздел печатается сразу после последней записи в группе, которая подвергается обработке. Этот раздел будет печататься ТОЛЬКО в случае, если свойство GroupTotalFields для элемента данных установлено, а элемент данных соответственно отсортирован.
TransportHeader	Промежуточный заголовок. Этот раздел печатается в верхней части страницы для элемента данных, вывод на печать которого продолжается с предыдущей страницы. Печатается после раздела Header.
TransportFooter	Промежуточный нижний колонтитул. Этот раздел печатается в нижней части страницы для элемента данных, вывод на печать которого будет продолжено на следующей странице. Печатается перед разделом Footer

Для печати итогов вложенного элемента данных Sales Line будут использоваться оба раздела: GroupFooter и Footer.

Итоги по полям Quantity и Amount для каждого **No.** (поле **No.** используется для группировки) будут выводиться в раздел GroupFooter.

Когда будут обработаны все записи элемента данных Sales Line, будет выведен раздел Footer с общими итогами.

Подведем итог по данному примеру:

1. Для каждой записи элемента данных Customer открывается выборка по элементу данных Sales Line.

2. Когда значение поля **No.** изменяется, выводится раздел GroupFooter.

3. Когда обработаны все записи элемента данных Sales Line, выводится раздел Footer.

Раздел Body элемента данных Customer выводиться перед тем, как будут выведены секции вложенного элемента данных, раздел Footer является последним. Таким образом, данный раздел можно использовать для вывода по клиентам.

Итоги по полям Quantity и Amount для всех товаров, заказанных определенным покупателем, будут помещены в раздел GroupFooter элемента данных Sales Line. Общий итог по полю Amount будет помещен в раздел Footer элемента данных Sales Line и будет информировать об общей сумме заказов (итог по полю Quantity также поддерживается системой, но оно не очень информативно, т.к. оно будет содержать общее количество для всех видов товаров.)

Создание Разделов

Следующим шагом будет создание разделов отчета.

Чтобы создать разделы:

DRAFT

1. Откройте в Report Designer отчет, над которым вы работали ранее.

2. Щелкните Вид, Sections.

3. Выберите элемент данных Sales Line.

4. Щелкните Правка, Создать и добавьте раздел GroupFooter. В данном разделе будет печататься итоговая информация по каждому товару.

5. Щелкните Правка, Создать и добавьте раздел Footer для элемента данных Sales Line. В данном разделе будет печататься итог по клиенту.

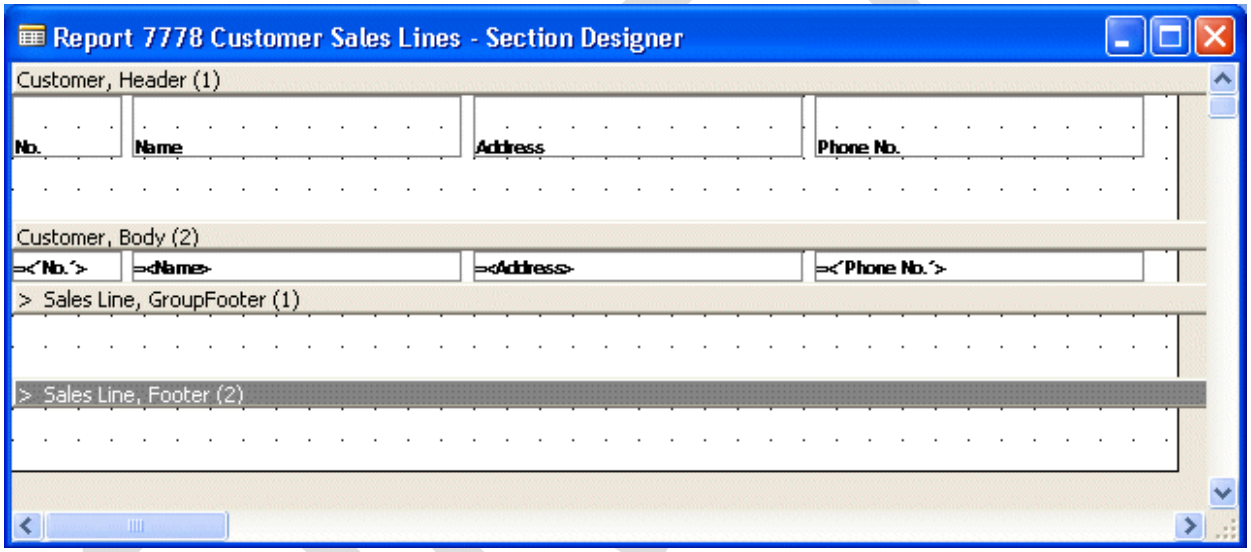
В данном отчете ничего не будет печататься в разделе Body элемента данных Sales Line.

Эту секцию нужно удалить:

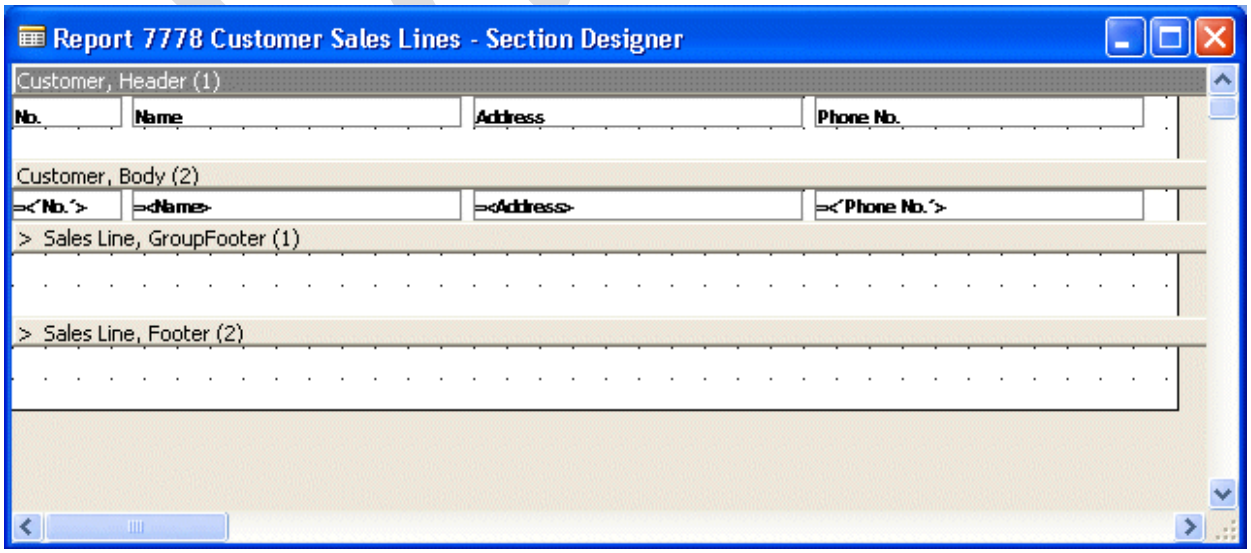
1. Щелкните по заголовку раздела Body элемента данных Sales Line, затем щелкните Правка, Удалить (F4) или нажмите клавишу DELETE.

2. Система попросит подтвердить намерение удалить данный раздел.

3. Выберите надписи для элемента данных Sales Line, которые ранее были добавлены в раздел Customer Header и удалите их. Section Designer должен выглядеть следующим образом:



4. Измените размер по вертикали надписей в разделе Header, а также размер самого раздела. Section Designer должен выглядеть следующим образом:



Теперь настало время отредактировать разделы Footer и убедится, что они отображают правильную информацию.

Чтобы отредактировать разделы Footer:

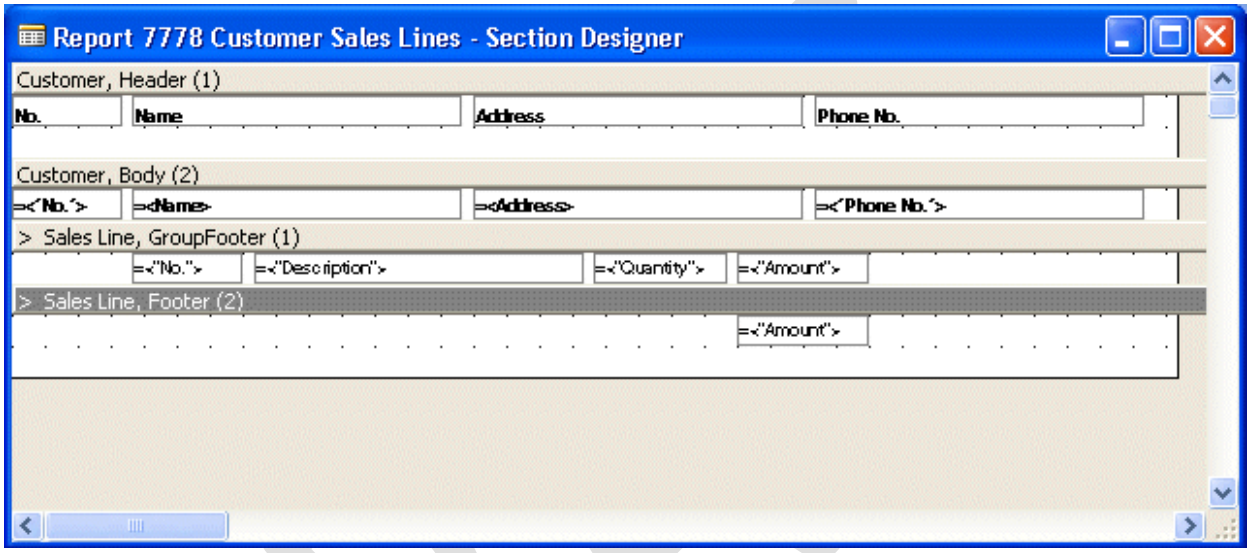
1. Выберите раздел GroupFooter элемента данных Sales Line и щелкните Вид, Field Menu, чтобы открыть окно ***Field Menu***.

2. В окне ***Field Menu*** выбрать поля **No.**, **Description**, **Quantity** и **Amount**.

3. Дважды щелкнуть по разделу GroupFooter элемента данных Sales Line, чтобы вставить эти поля. Удалите надписи и измените, размер раздела по вертикали.

4. Выберите раздел Footer элемента данных Sales Line.

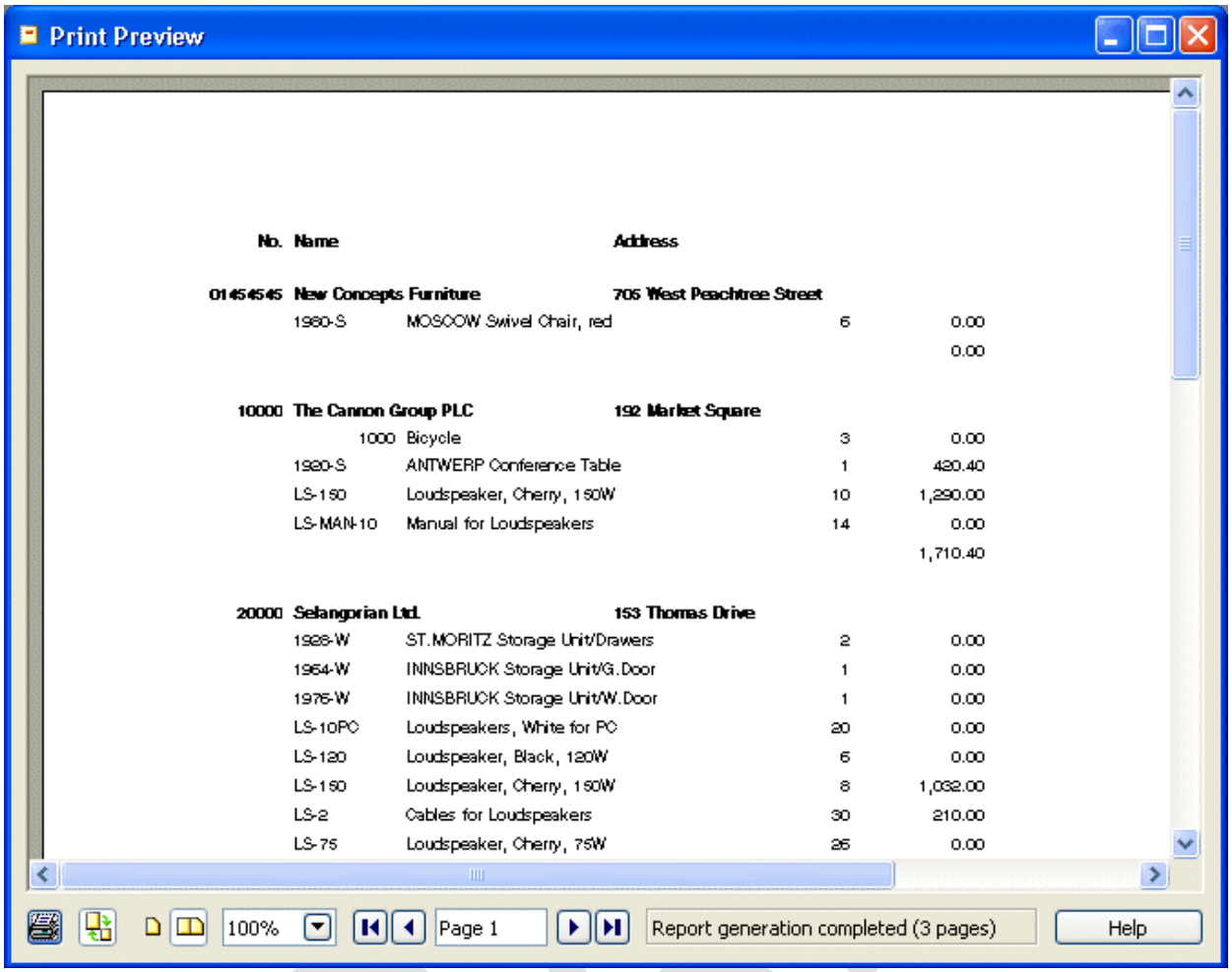
5. В окне ***Field Menu*** выберите поле **Amount** и вставьте его в раздел footer так, чтобы оно оказалось под полем **Amount** раздела GroupFooter. Удалите надпись. Не изменяйте размер раздела, чтобы перед следующим клиентом было пустая строка. Section Designer должен выглядеть следующим образом:



6. Выберите надпись No. в разделе Customer Header и откройте окно ***Properties*** (SHIFT+F4), затем установите в свойство *HorzAlign* значение Right.

7. Сохраните, закройте и запустите отчет.

8. Отчет должен выглядеть следующим образом:



Print Preview

No.	Name		Address	
01454545	New Concepts Furniture		705 West Peachtree Street	
	1980-S	MOSCOW Swivel Chair, red	6	0.00
				0.00
10000	The Cannon Group PLC		192 Market Square	
	1000	Bicycle	3	0.00
	1980-S	ANTWERP Conference Table	1	420.40
	LS-150	Loudspeaker, Cherry, 150W	10	1,290.00
	LS-MAN-10	Manual for Loudspeakers	14	0.00
				1,710.40
20000	Selangorian Ltd.		153 Thomas Drive	
	1928-W	ST.MORITZ Storage Unit/Drawers	2	0.00
	1964-W	INNSBRUCK Storage Unit/G.Door	1	0.00
	1976-W	INNSBRUCK Storage Unit/W.Door	1	0.00
	LS-10PC	Loudspeakers, White for PC	20	0.00
	LS-120	Loudspeaker, Black, 120W	6	0.00
	LS-150	Loudspeaker, Cherry, 150W	8	1,032.00
	LS-2	Cables for Loudspeakers	30	210.00
	LS-75	Loudspeaker, Cherry, 75W	26	0.00

Над этим отчетом еще надо поработать в части функциональности и внешнего вида. Например, нужно решить, куда поместить заголовки столбцов для вложенного элемента данных.

Тем не менее, логика отчета работает верно: для каждого клиента выведен список товаров с указанием общего заказанного количества и суммой, также рассчитывается сумма заказов по клиенту.

Еще одно улучшение, которое необходимо данному отчету – это общий итог по всем клиентам. Однако чтобы сделать это нужно будет написать код на языке C/AL в триггере отчета. В следующих разделах будет рассказано, как это сделать.

14.2 Триггеры в Отчетах

Хотя система определяет и обрабатывает большинство событий определенным образом, часть событий вызывает триггеры. Триггеры – это предопределенные функции, которые выполняются при наступлении некоторых событий. Изначально эти функции пусты. Добавление C/AL кода в триггеры позволяет изменять поведение системы, так чтобы оно соответствовало поставленным задачам. В отчетах триггеры обычно используются для выполнения вычислений, а также для указания должен ли определенный раздел выводиться на печать или нет. Это может зависеть, например, от значения в поле или параметров заданных пользователем на форме запроса. Но самым большим преимуществом триггеров является то, что они позволяют управлять процессом выбора и получения данных более точно и эффективно, чем при использовании свойств.

Отчеты могут содержать следующие триггеры:

Триггеры Отчетов

Эти триггеры применяются непосредственно к отчету:

Триггер	Когда выполняется
OnInitReport	Когда отчет загружен
OnPreReport	Перед тем как отчет будет запущен, но после того как была запущена форма запроса
OnPostReport	После того как работа отчета завершена (кроме ручной остановки или использования функции Break)
OnCreateHyperlink	При создании URL для отчета (например, при отправке по электронной почте или на рабочий стол)
OnHyperlink	При нажатии по гиперссылке, выполняется после триггера *OnInitReport*.

Триггеры Элементов Данных

Следующие триггеры применяются к каждому элементу данных в отчете:

Триггер	Когда выполняется
OnPreDataItem	Перед тем как данные начнут обрабатываться, но после того как была проинициализирована переменная, связанная с данным элементом данных.
OnAfterGetRecord	После получения записи из таблицы
OnPostDataItem	После того как была обработана последняя запись в элементе данных.

Триггеры Разделов

Эти триггеры применяются к каждому разделу элемента данных:

Триггер	Когда выполняется
OnPreSection	Перед обработкой раздела.
OnPostSection	После обработки раздела, но перед тем как он будет напечатан.

Дополнительную информацию о данных триггерах см. в онлайновой справке *C/SIDE Reference Guide*.

14.3 Создание Отчета с Использованием Виртуальной таблицы

В среду C/SIDE включен определенный набор виртуальных таблиц, таких как ***Integer*** или ***Date***. Виртуальные таблицы рассматривались в главе ГЛАВА 8 СПЕЦИАЛЬНЫЕ ТАБЛИЦЫ C/SIDE. В данном разделе будет приведен пример использования таблицы ***Date*** в отчете.

Использование Таблицы Date Table

Таблица ***Date*** состоит из пяти полей: **Period Type**, **Period Start, Period End**, **Period No.** и **Period Name**.

Поле **Period Type** может принимать значения: Date, Week, Month и т.д. В поле **Period Start** указывается дата начала каждого периода, а в поле **Period End** – последняя дата периода. (В поле **Period End** указываются закрывающие даты.)

DRAFT

В данном примере таблица ***Date*** используется для печати информации из таблицы ***Cust. Ledger Entry***. Для каждой даты в выбранном пользователе интервале отчет просуммирует все операции на эту дату.

Отчет выведет список различных типов документов, число документов каждого типа и общее число документов на каждую дату. Также в отчете будет показана итоговая сумма на каждую дату. В конце отчета будет выведена общая сумма по всем операциям, а также общее количество всех операций.

Отчет можно создать, группируя операции таблицы ***Cust. Ledger Entry***, но в ней нет ключа, начинающегося с поля, содержащего дату учета (posting date). Для одного отчета создавать специальный ключ не целесообразно, так как это замедлит выполнение всех транзакций выполняющихся в данной таблице. Это негативно скажется на каждой операции продажи.

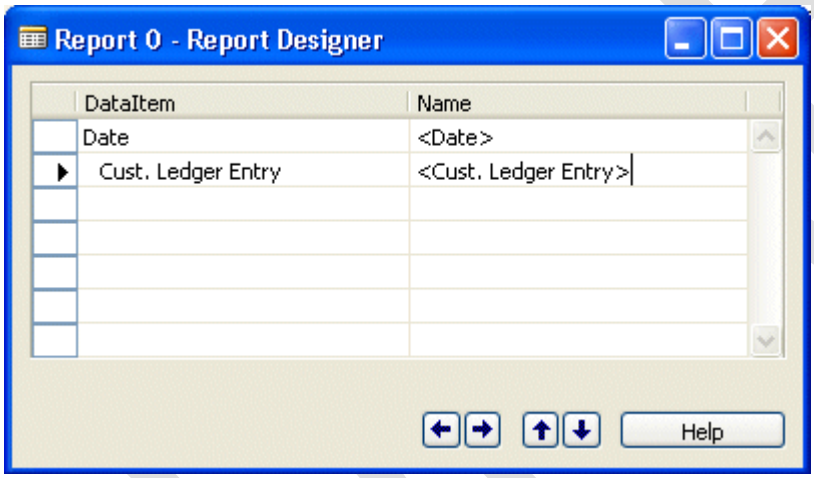
Определение Модели Данных

Модель данных содержит два элемента данных: один основан на таблице ***Date***, второй на таблице ***Cust. Ledger Entry***.

Чтобы определить модель данных:

1. Откройте Report Designer и создайте новый постой отчет с двумя элементами данных: ***Date*** и ***Cust. Ledger Entry***.

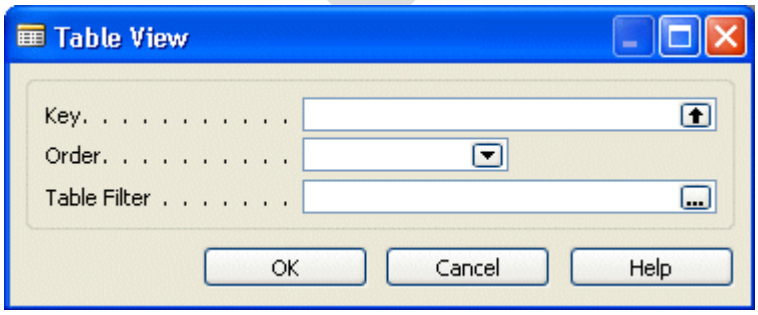
2. Выберите элемент данных Customer Ledger Entry и щелкните один раз кнопку со стрелкой вправо, чтобы задать вложенность:



Вложенность гарантирует, что система будет выполнять поиск по таблице ***Customer Ledger Entry,*** только если в таблице ***Date*** найдена дата, внутри диапазона определенного пользователем.

3. Откройте окно ***Properties*** (SHIFT+F4) для элемента данных Date.

4. Щелкните кнопку AssistButton в свойстве *DataItemTableView*, чтобы открыть окно ***Table View***:



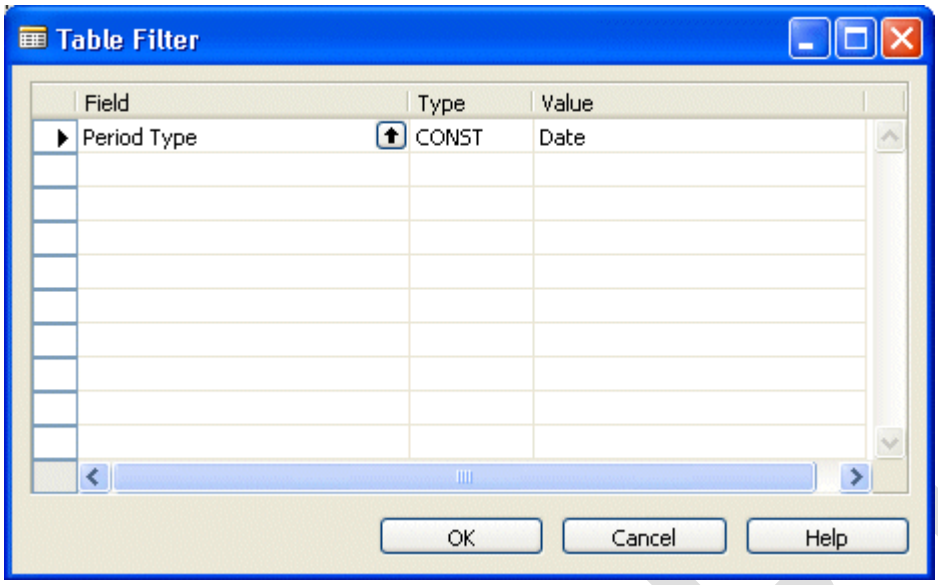
5. Щелкните кнопку AssistButton в поле **Table Filter**, чтобы открыть окно ***Table Filter***.

6. В окне ***Table Filter*** установите фильтр, чтобы отбирались только те записи, у которых Period Type = Date.

Важно

Это важный шаг, так иначе в выборку попадут записи для типов Weeks, Quarters, Months и Years.

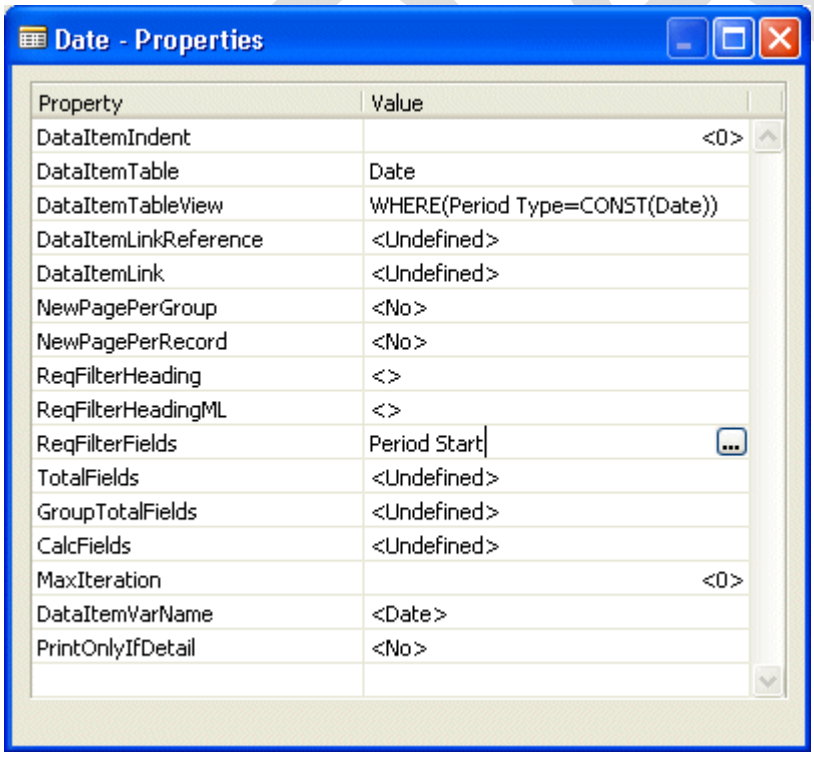
Окно ***Table Filter*** должно выглядеть следующим образом:



7. Щелкните кнопку OK в окнах ***Table Filter*** и ***Table View***.

8. В окне ***Properties*** элемента данных Date укажите в свойстве *ReqFilterFields* значение *Period Start*. Это позволит пользователю вводить диапазон дат в форму запроса перед запуском отчета. Эта информация будет использоваться для фильтрации данных выводимых в отчет.

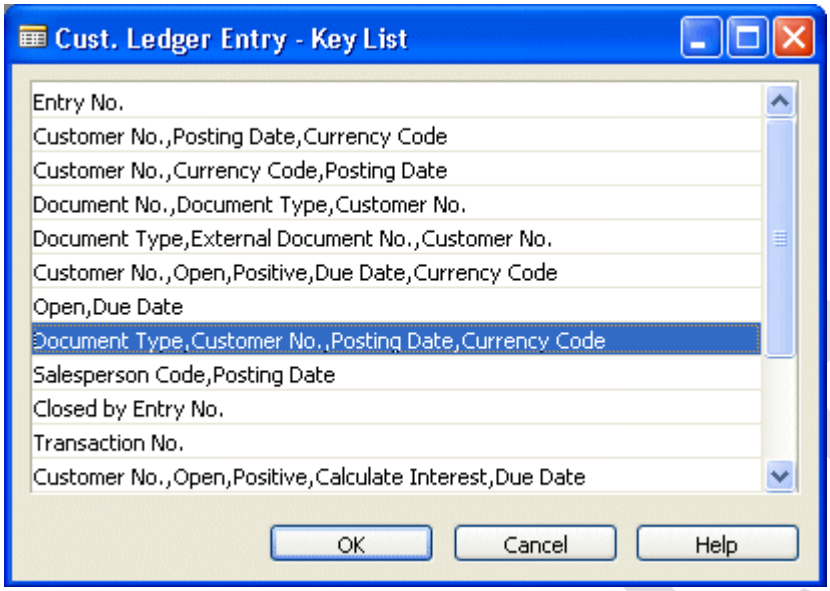
Окно ***Properties*** должно выглядеть следующим образом:



9. Открыть окно ***Properties*** элемента данных Cust. Ledger Entry. Щелкните кнопку AssistButton в поле *DataItemTableView,* чтобы открыть окно ***Table View***.

В поле **Key** щелкните кнопку Выбор, чтобы открыть окно ***Key List***.

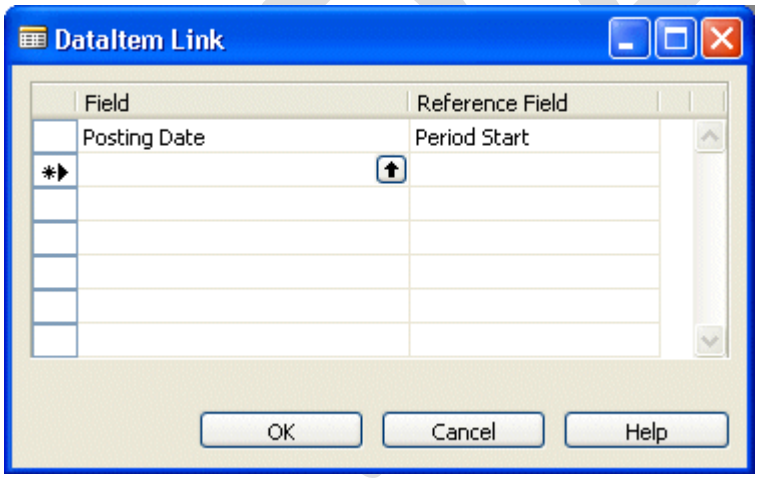
Для данного примера подойдет ключ, который содержит поле **Document Type**, так как данное поле будет использоваться для группировки. Выбранный ключ содержит данное поле на первой позиции, остальные поля не важны, т.к. сами записи печататься не будет, поэтому нет необходимости сортировать их каким-либо определенным образом.



10. Убедитесь, что свойство *DataItemLinkReference* указывает на элемент данных Date (по умолчанию).

11. В свойстве *DataItemLink* нужно указать поля, которые будут связывать между собой два элемента данных. Щелкните кнопку AssistButton в свойстве *DataItemLink*, чтобы открыть окно ***DataItem Link***.

В поле **Field** укажите поле **Posting Date** из элемента данных Cust. Ledger Entry, а в поле **Reference Field** укажите **Period Start** из таблицы Date.



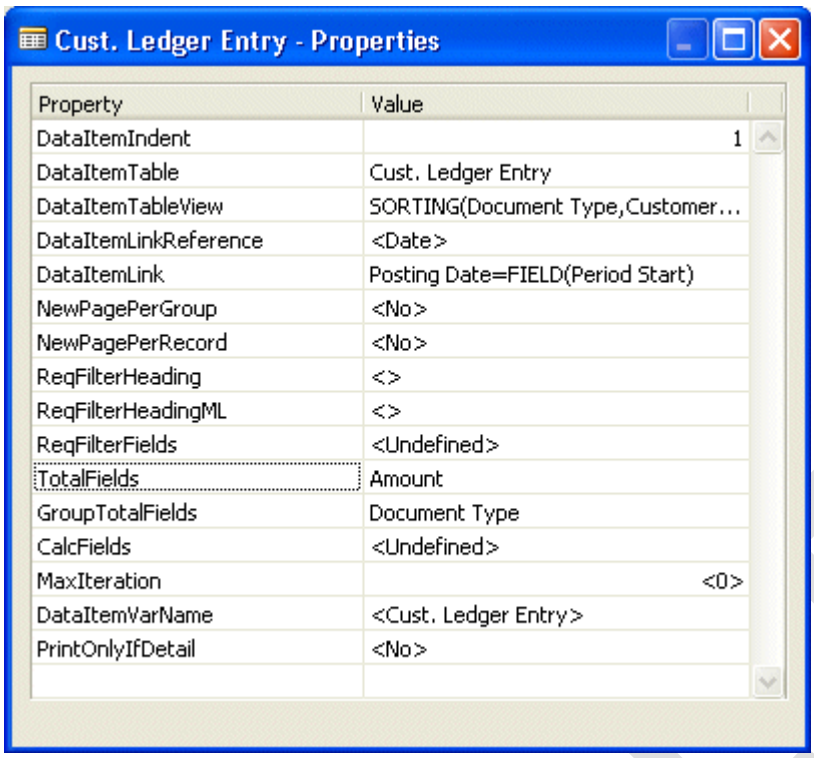
12. В свойстве *GroupTotalFields* укажите поле **Document Type**.

Это укажет системе сгруппировать операции из таблицы ***Cust. Ledger Entry*** по типам документов: Payment, Invoice, Credit Memo, Finance Charge Memo, Reminder или Refund.

13. В свойстве *TotalFields* укажите поле **Amount**.

Это укажет системе просуммировать значения поля **Amount** по типам документом, для заданной даты.

Окно ***Properties*** элемента данных Cust. Ledger Entry должно выглядеть следующим образом:



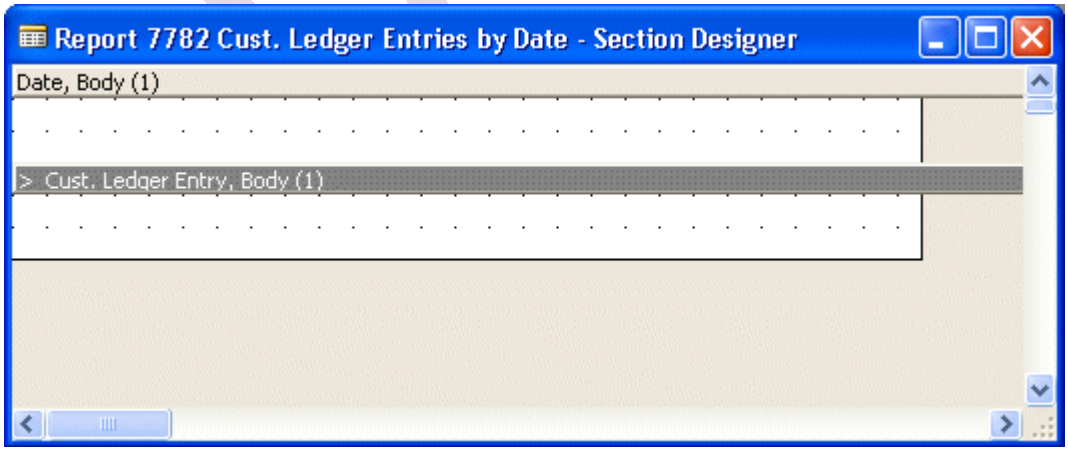
Итак, отчет будет работать следующим образом:

- На форме запроса отчета можно будет указать интервал дат.
- Отчет обрабатывает записи таблицы ***Date***, применив к ней строгий фильтр по полю **Period Type,** указав, что должны выбираться только записи с типом *Date*. Если указан диапазон дат, то в отчет попадают только даты из указанного диапазона, иначе используются все даты.
- Для каждой даты выбираются операции из элемента данных Cust. Ledger Entry, у которых дата учета совпадает с выбранной датой (записью таблицы *Date)*.
- Записи элемента данных Cust. Ledger Entry группируются по полю **Document Type**, для поля **Amount** поддерживаются итоги.

Создание Разделов

Чтобы создать разделы:

1. Щелкните Вид, Sections, чтобы открыть **Section Designer**:



Для каждого элемента данных уже созданы разделы body.

2. Щелкните Правка, Создать (F3) и вставьте раздел Header для элемента данных Date. В данном разделе будут содержаться надписи столбцов отчета.

DRAFT

3. Вставьте два раздела GroupFooter для элемента данных Cust. Ledger Entry. Так как создано два раздела GroupFooter, то управлять их появлением можно отдельно. Оба раздела содержат итоговую информацию по группам, созданным для этого элемента данных.

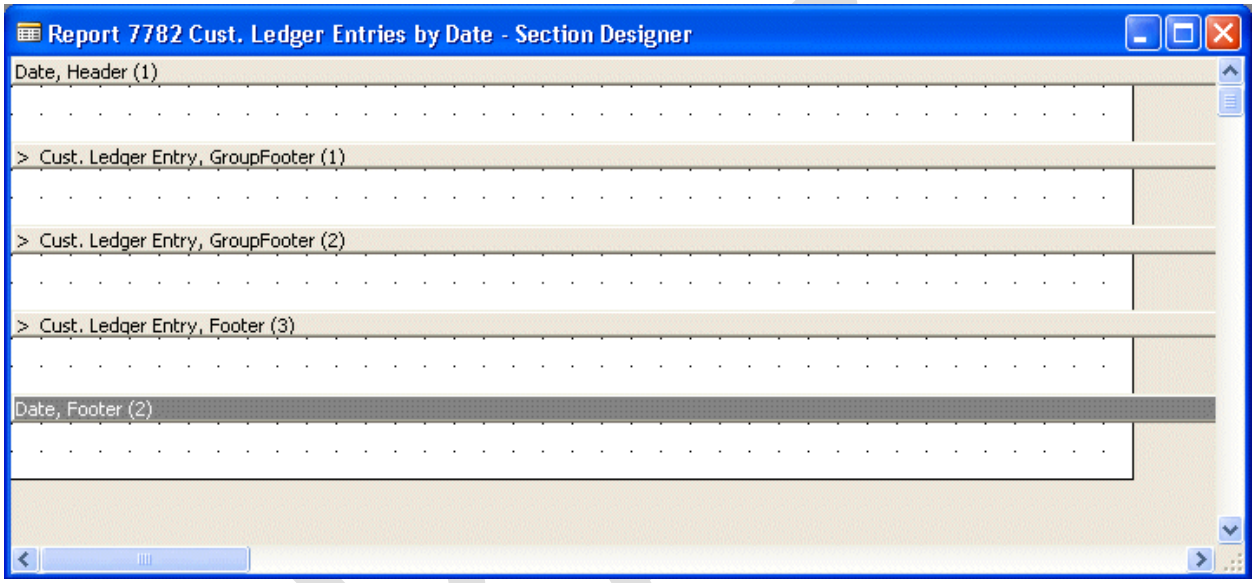
Причина, по которой используется такая конструкция, а также как ее использовать будет объяснено далее.

4. Вставьте раздел Footer для элемента данных Cust. Ledger Entry. Данный раздел будет использоваться для отображения промежуточных итогов.

5. Вставьте раздел Footer для элемента данных Date. Данный раздел будет использоваться для отображения общих итогов в конце отчета.

6. Так как ни для одного элемента данных не нужны разделы Body, то их можно удалить.

Section Designer должен выглядеть следующим образом:



Добавление Полей в Отчет

На следующем шаге выполним добавление требуемых полей в различные разделы отчета.

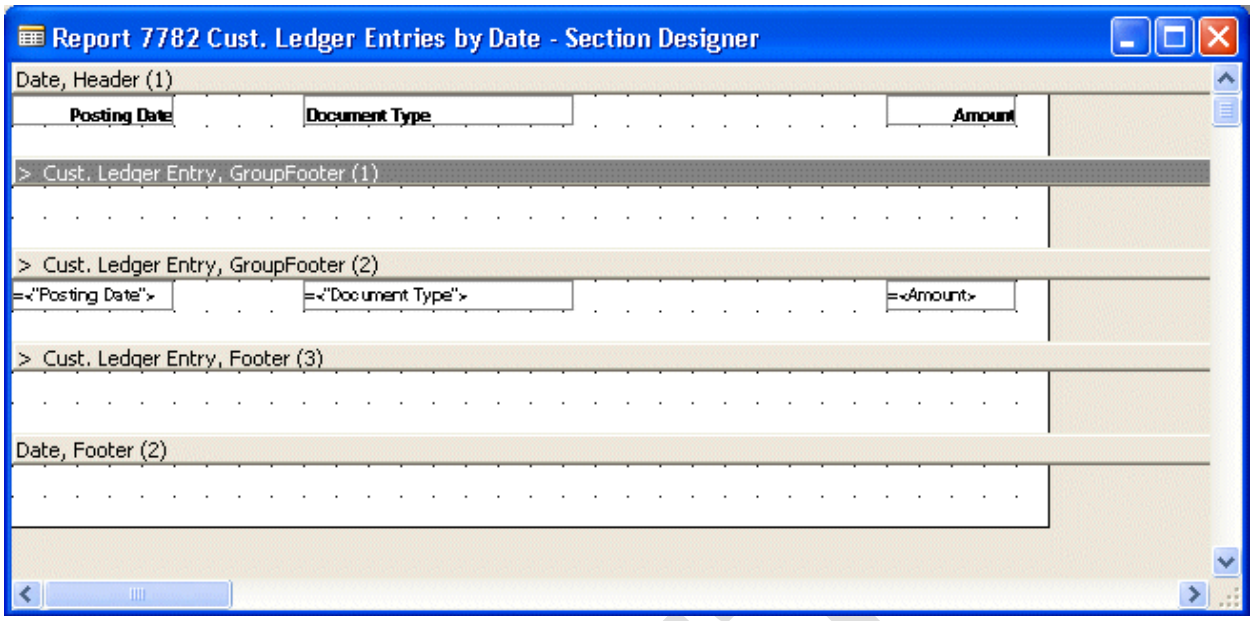
Чтобы добавить поля в отчет:

1. В окне Section Designer выберите раздел **Cust. Ledger Entry, GroupFooter (2),** затем щелкните Вид, Field Menu, чтобы открыть окно ***Field Menu***.

2. В окне ***Field Menu*** выберите поля **Posting Date**, **Document Type** и **Amount**.

3. В окне Section Designer дважды щелкните по разделу **Cust. Ledger Entry, GroupFooter (2)**, чтобы добавить данные поля.

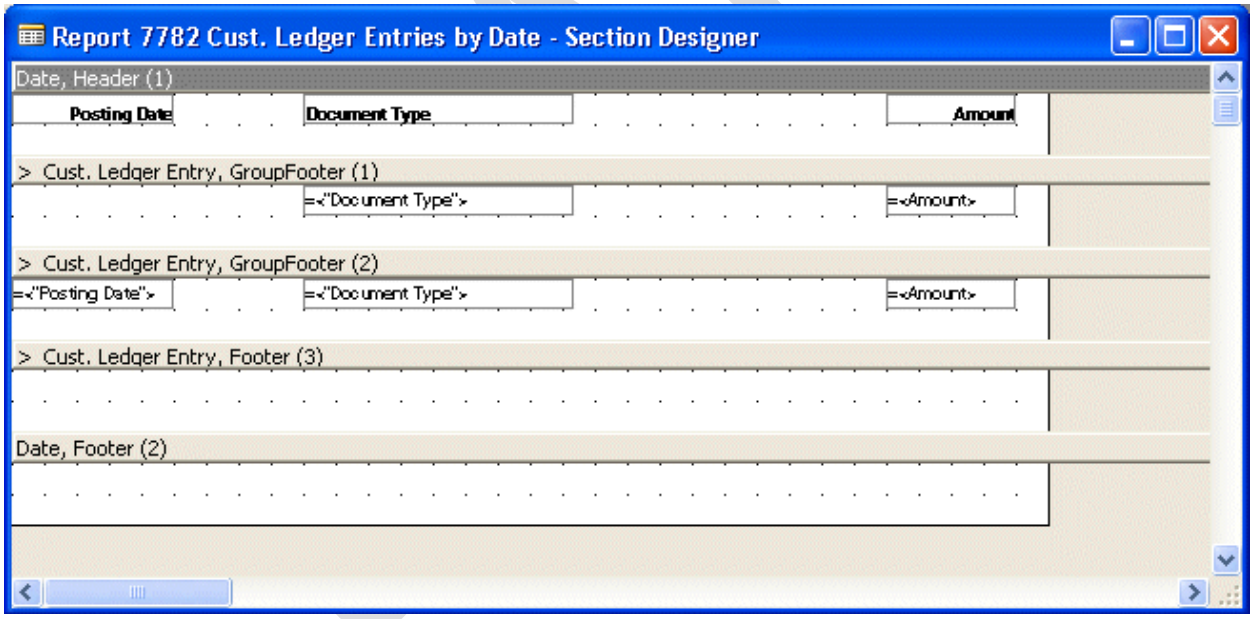
4. Переместите надписи в раздел Header и выровняйте поля так, чтобы они выглядели следующим образом:



5. В окне Section Designer выберите раздел **Cust. Ledger Entry, GroupFooter (1)** и добавьте поля **Document Type** и **Amount**.

6. Удалите надписи для полей добавленных в раздел **Cust. Ledger Entry, GroupFooter (1)**.

7. Выровняйте поля так, чтобы они выглядели следующим образом (возможно, потребуется увеличить редактируемую область):



Как видно, отчет пока далек от завершения. Еще требуется добавить поля, которые:

- Содержат количество документов каждого типа, учтенных на заданную дату.
- Рассчитывают и отображают общее количество документов, учтенных на заданную дату.
- Рассчитывают и отображают итоговое количество документов, учтенных в охватываемом периоде.
- Рассчитывает и отображает общую сумму по документам, учтенным на заданную дату.
- Рассчитывает и отображает итоговую сумму по документам, учтенным в охватываемом периоде
- Отображает диапазон, который ввел пользователь для фильтрации информации в отчете.

Чтобы реализовать указанные требования, нужно добавить некоторое количество кода на языке C/AL в триггерах, определить несколько переменных и добавить несколько соответствующих полей.

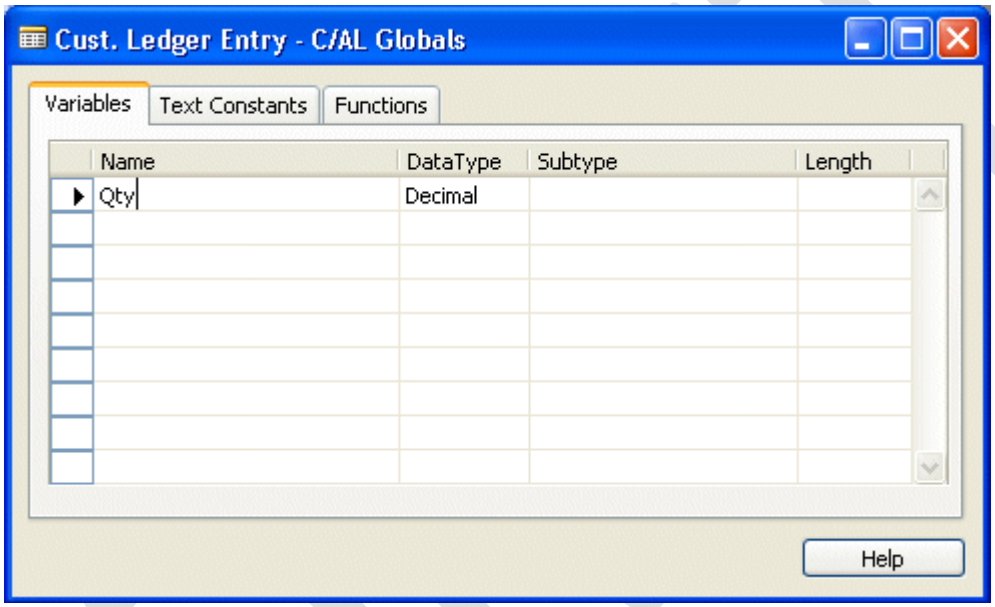
Подсчет Количества Записей

Суммирование значений по полю **Amount** элемента данных Cust. Ledger Entry выполняется с помощь настройки свойств элемента данных. Эта настройка была выполнена на шаге 13, когда с помощью свойства *TotalFields* было указано, что система должна рассчитывать итоги по полю **Amount** таблицы ***Cust. Ledger Entry***. Эти итоги отдельно считаются для каждой даты и служат основой для расчета общего итога, который будет рассчитан позже.

К сожалению, таким образом нельзя рассчитать количество записей, т.к. у элемента данных нет поля, которое содержит такую информацию. Однако каждая запись таблицы ***Cust. Ledger Entry*** соответствует одной операции, а это означает, что количество операций можно определить подсчитав количество записей.

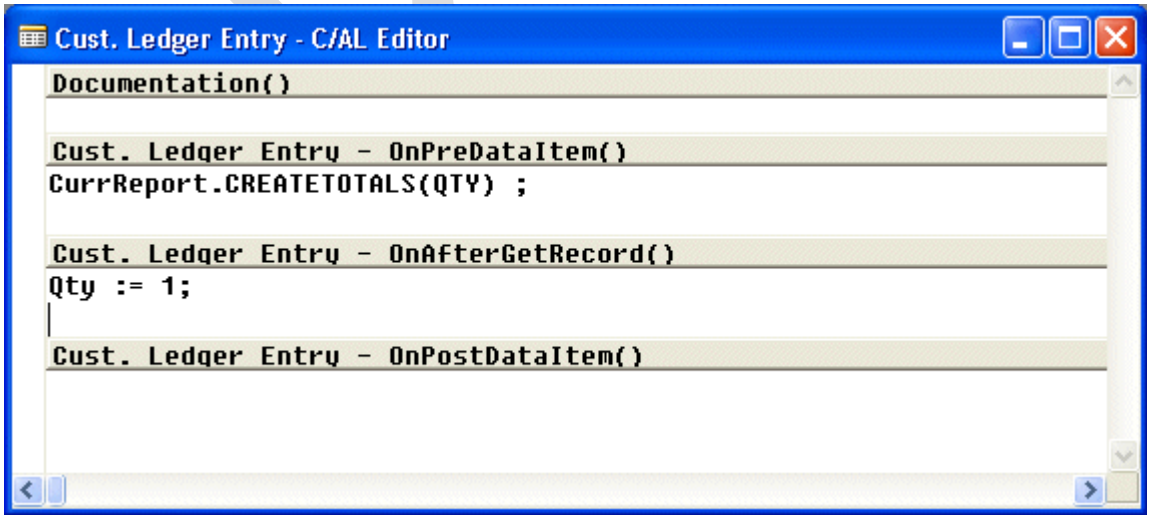
Чтобы рассчитать общее количество операций на каждую дату нужно:

1. Создать глобальную переменную (здесь - Qty) и указать ей тип данный *Decimal*:



Выбрать элемент данных Cust. Ledger Entry в Report Designer и щелкнуть Вид, C/AL Code (F9), чтобы открыть окно C/AL Editor.

2. Добавить следующий C/AL код в триггеры Cust. Ledger Entry:



Функция в триггере *OnPreDataItem* поддерживает итоги для переменной Qty таким же образом, как свойство *TotalFields* поддерживает итоги для полей записи. Выражение в триггере *OnAfterGetRecord* присваивает переменной Qty значение 1 каждый раз при получении записи из базы.

Функция CREATETOTALS используется для поддержки итогов для групп и общих итогов элемента данных. Так как элемент данных сгруппирован по полю **Document Type**, переменная Qty будет содержать общее количество документов каждого типа, каждый раз при печати раздела **Cust. Ledger Entry GroupFooter**. При печати раздела Footer переменная Qty будет содержать общее количество всех операций (все записи на заданную дату).

В качестве параметра для функции CREATETOTALS следует использовать переменные с типом данных Decimal, т.к. данная функция обычно используется для сложения значений содержащих суммы. По этой причине переменная Qty была объявлена как тип Decimal, а не как Integer (что было бы логичнее).

Теперь нужно добавить несколько полей в отчет, чтобы отображать итоги, рассчитанные с помощью приведенного выше кода:

1. Открыть Section Designer, открыть панель инструментов Toolbox.

2. Добавить в раздел **Cust. Ledger Entry, GroupFooter (1)** текстовую зону и открыть для нее окно ***Properties***.

3. Щелкнуть кнопку AssistButton в свойстве *SourceExpr*, чтобы открыть окно **C/AL Symbol Menu**.

4. Выбрать переменную Qty.

5. Добавить в раздел Header элемент управления Надпись, открыть окно ***Properties*** для надписи и указать в свойстве *Caption* значение *Qty*. Убедитесь, что в свойстве *HorzAlign* установлено значение *Right*.

Переменная Qty объявлена как decimal, поэтому нужно изменить способ форматирования текстовой зоны Qty так, чтобы отображались только целые значения (не отображались знаки после запятой). При указании в свойстве SourceExpr переменной типа Decimal, для формата текстовой зоны было установлено значение по умолчанию: <2:2>.

Чтобы изменить форматирование текстовой зоны Qty:

1. Выберите текстовую зону в Section Designer и откройте окно ***Properties***.

2. В свойстве *DecimalPlaces* укажите значение 0:0. Больше в поле не будут отображаться десятичные значения.

Следующим шагом будет добавление поля, которое содержит общее количество документов каждого типа на заданную дату.

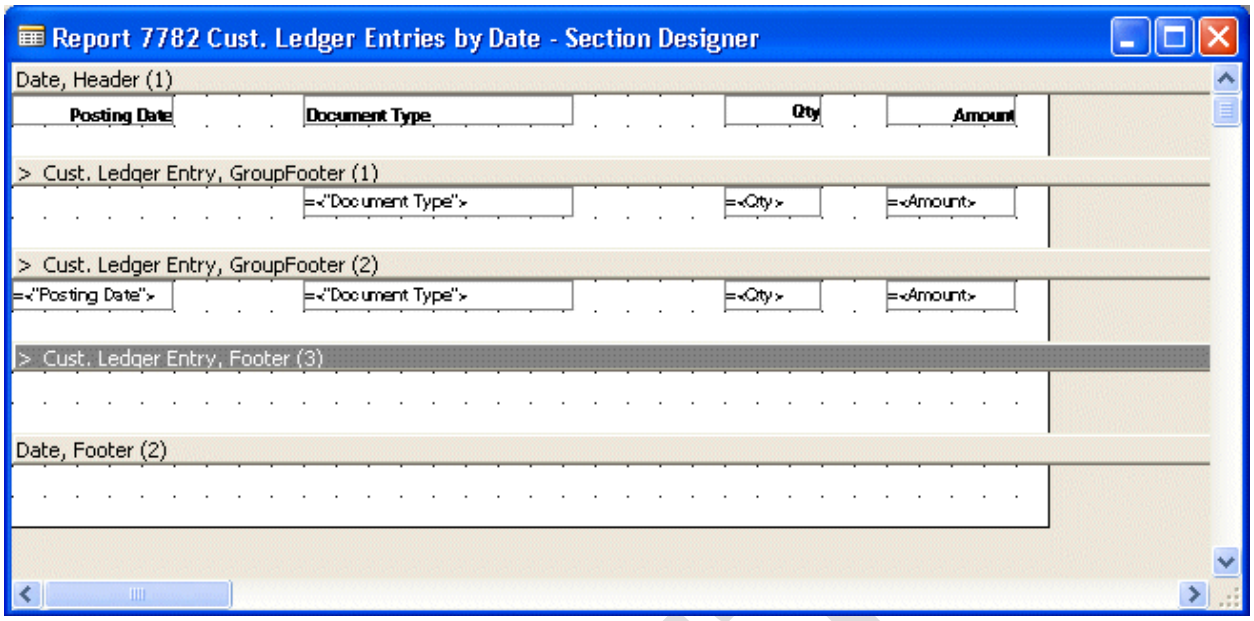
1. Скопируйте текстовую зону Qty, которую добавили в раздел **Cust. Ledger Entry, GroupFooter (1)**.

2 Вставьте ее в раздел **Cust. Ledger Entry, GroupFooter (2)**.

Эта текстовая зона будет отображать количество документов каждого типа на заданную дату.

Теперь Section Designer должен выглядеть следующим образом:

DRAFT



Теперь отчет содержит поля, в которых выводится количество документов каждого типа на каждую дату.

Расчет Общего Количества Операций и Сумма за День

Следующим шагом в создании отчета будет вывод промежуточных итогов, в которых будет указано общее количество документов за день и сумма по документам за день.

Для этих целей была создан раздел **Cust. Ledger Entry, Footer (3)**.

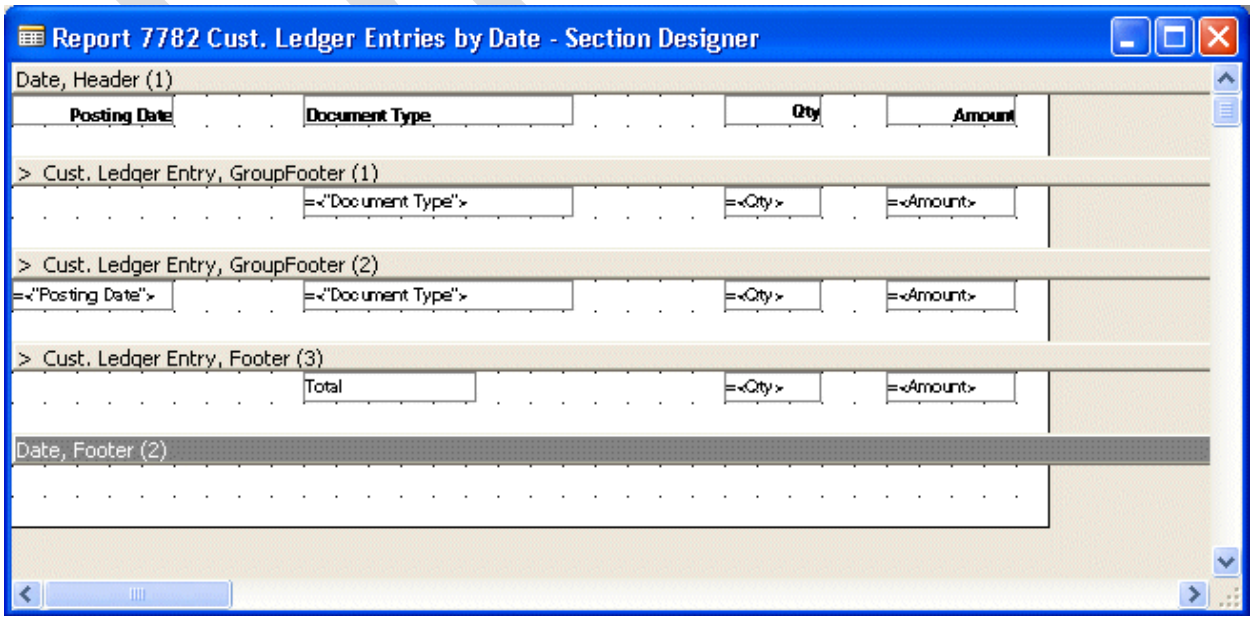
Чтобы рассчитать промежуточные итоги:

1. Скопируйте поля **Qty** и **Amount** из раздела **Cust. Ledger Entry, GroupFooter (2)** в раздел **Cust. Ledger Entry, Footer (3)**.

Теперь в данных полях будут отображаться общие итоги за день.

2. Добавьте в раздел **Cust. Ledger Entry, Footer (3)** надпись и откройте для нее окно ***Properties***, затем укажите в свойстве *Caption* значение *Total*.

Теперь Section Designer должен выглядеть следующим образом:



Печать Даты Только Для Первой Итерации

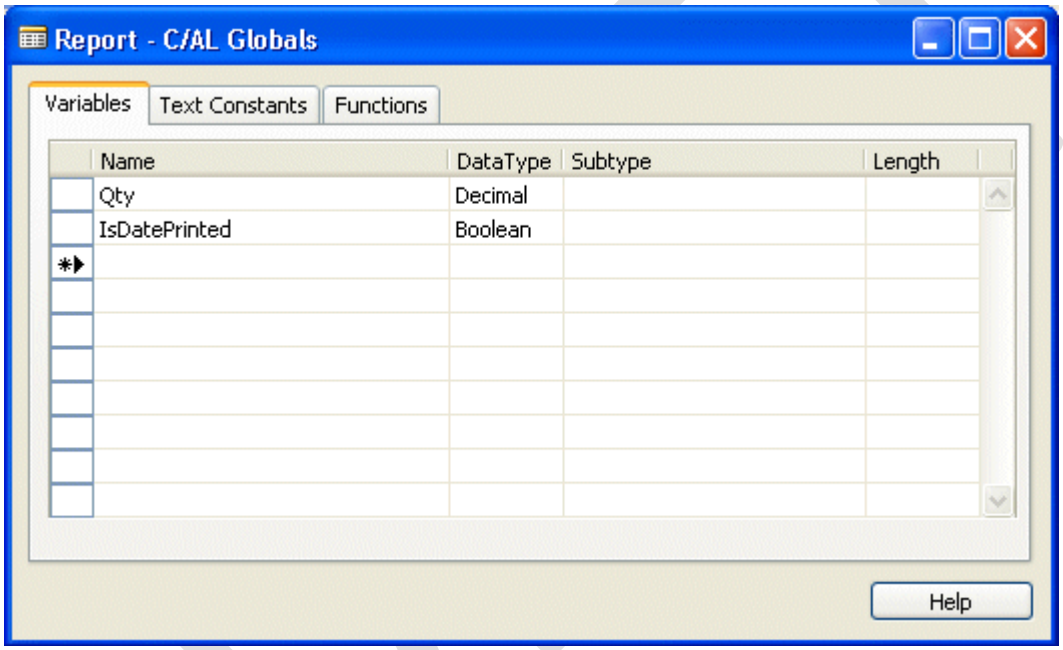
При каждой смене даты в элементе данных Date, если для этой даты существуют операции в элементе данных Cust. Ledger Entry, нужно выводить дату на печать. Конечно, можно добавить раздел Date и печатать дату в нем (поле **Period Start**), но это означает, что дата будет печататься на отдельной строке. Вместо этого нужно, чтобы дата была напечатана вместе с остальной информацией первой строки элемента данных Cust. Ledger Entry.

Вариантом решения может стать вывод поля **Date** в элементе данных Cust. Ledger Entry. К сожалению это влечет свои проблемы. Если добавить поле к разделу GroupFooter, то дата будет печататься в каждой строке. Хотя это и является простым способом решения проблемы, напечатанный отчет будет не слишком привлекательным из-за перегруженности избыточной информацией.

Лучшим решение будет использовать два раздела GroupFooter для элемента данных. Один из них будет включать поле **Date**, а второй нет. Также потребуется управлять процессом вывода. В этом заключается причина, по которой было создано два раздела GroupFooter.

Чтобы добавить дату и контролировать, когда должны использоваться разные разделы следует:

1. Создать глобальную переменную IsDatePrinted с типом данных Boolean.



2. В триггер *OnPreDataItem* элемента данных Cust. Ledger Entry добавить следующий C/AL код:

```
IsDatePrinted := FALSE
```

Этот код инициализирует переменную IsDatePrinted значением FALSE перед каждым обработкой элемента данных.

3. Добавьте следующий код на языке C/AL в триггер *OnPreSection* раздела **Cust. Ledger Entry, GroupFooter (1)**:

```
IF IsDatePrinted THEN
  CurrReport.SHOWOUTPUT(TRUE)
ELSE
  CurrReport.SHOWOUTPUT(FALSE);
```

4. Добавьте следующий код на языке C/AL в триггер *OnPreSection* раздела **Cust. Ledger Entry, GroupFooter (2)**:

```
IF IsDatePrinted THEN
```

```
    CurrReport.SHOWOUTPUT(FALSE)
  ELSE BEGIN
    CurrReport.SHOWOUTPUT(TRUE);
    IsDatePrinted := TRUE;
  END
```

Эти два фрагмента кода гарантируют, что:

- Перед каждой новой обработкой элемента данных Cust. Ledger Entry дата еще не печаталась.
- Для первой итерации выводится второй раздел GroupFooter (который содержит текстовую зону **Date**).
- Для остальных итераций выводится первый раздел GroupFooter (без текстовой зоны **Date**).

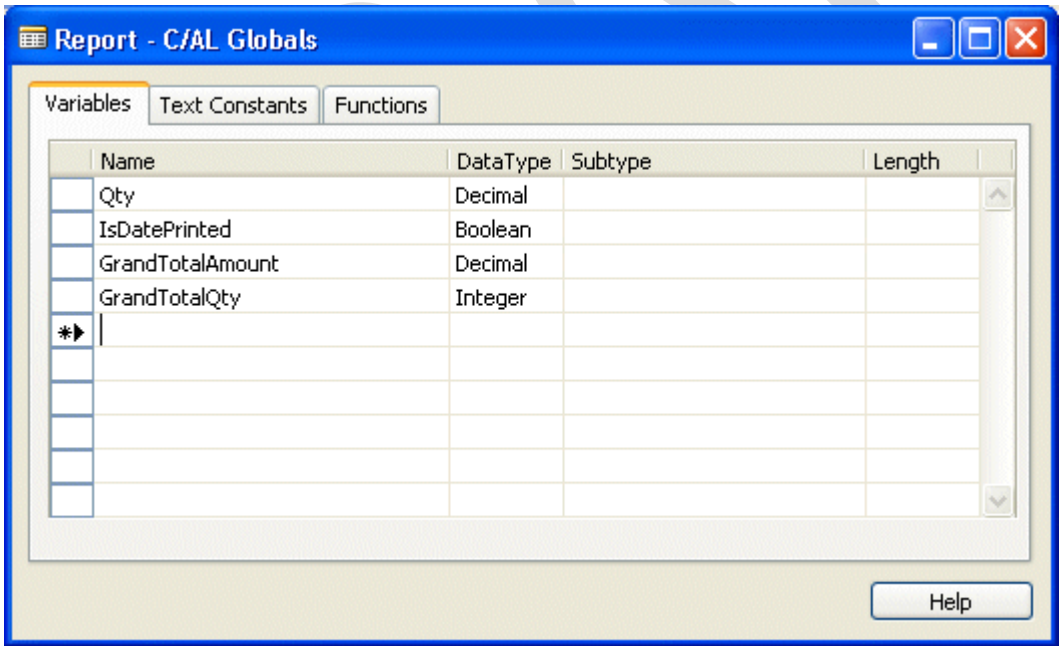
Расчет Общих Итогов по Количеству и Сумме

Чтобы завершить работу над отчетом, надо обеспечить вывод общих итогов по количеству документов и их сумме за весь период формирования отчета. Данная информация должна выводиться в разделе **Date Footer (2)**.

Следующим шагом будет расчет общих итогов по всем суммам, печатаемым в отчете. Так как эти суммы формируются на уровне элемента данных Cust. Ledger Entry, а не элемента данных Date, то использовать свойство *TotalFields* элемента данных Date для расчета итогов не получится.

Чтобы подсчитать общие итоги нужно:

1. Объявить две глобальные переменные: GrandTotalAmount и GrandTotalQty с типами данных Decimal и Integer соответственно:



2. Добавить следующий код в триггер *OnAfterGetRecord* элемента данных Cust. Ledger Entry:

```
GrandTotalQty := GrandTotalQty + 1
GrandTotalAmount := GrandTotalAmount + Amount
```

Первая строка увеличивает переменную GrandTotalQty на единицу при каждом извлечении записи из базы данных, во второй строке происходит увеличение переменной GrandTotalAmount на значение поля Amount, извлеченного из базы данных GrandTotalAmount.

3. Открыть Section Designer и добавить две текстовых зоны в раздел **Date, Footer (2)**.

4. Первая текстовая зона будет использоваться для расчета общего количества документов в отчете.

Поместите ее под поля **Qty** и откройте окно ***Properties*** данной текстовой зоны.

5. В свойстве *SourceExpr* щелкните кнопку AssistButton, чтобы открыть окно **C/AL Symbol Menu**.

6. Выберите переменную GrandTotalQty.

7. Вторая текстовая зона будет использоваться для расчета общего итога по суммам, отображаемым в отчете.

Поместите ее под другими полями **Amount** и откройте окно ***Properties*** данной текстовой зоны.

8. В свойстве *SourceExpr* щелкните кнопку AssistButton, чтобы открыть окно **C/AL Symbol Menu**.

9. Выберите переменную GrandTotalAmount.

В данной текстовой зоне будет отображаться итог по суммам в отчете.

10. Добавьте надпись в разделе **Date, Footer (2)** и поместите ее поде полем **Total** раздела **Cust. Ledger Entry, Footer (3)**.

11. Измените значение свойства *Caption* на Total.

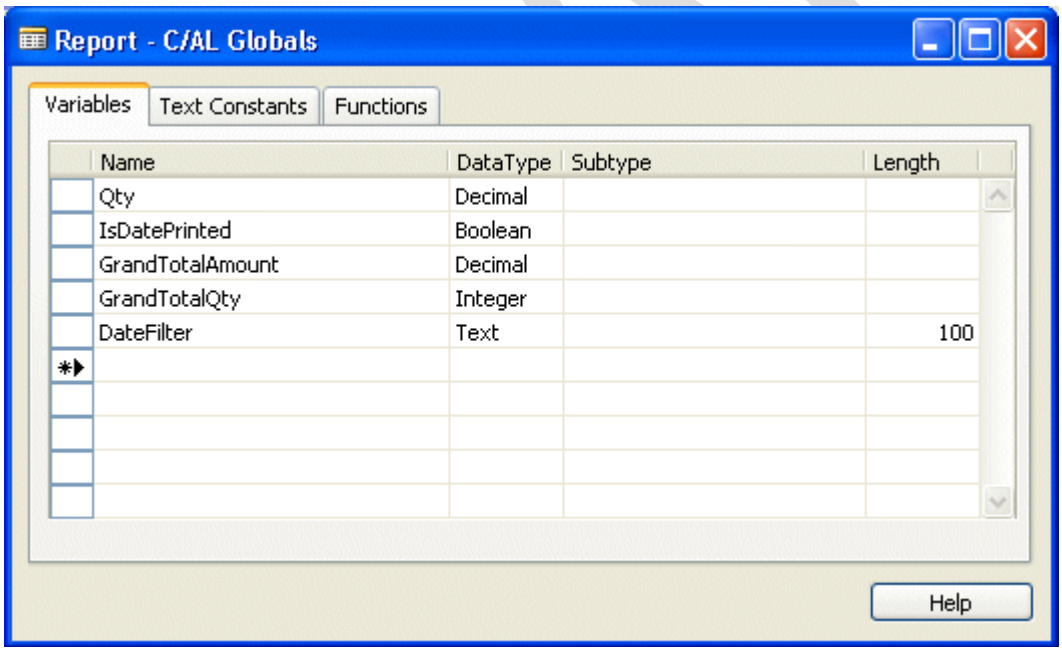
Теперь в разделе **Date, Footer (2)** содержаться общие итоги по отчету в целом.

Вывод на Печать Выбранного Диапазона Дат

Дата учета отображается в качестве заголовка в левой части отчета. Также было бы хорошо отображать в последней строке диапазон дат, указанный пользователем перед запуском отчета.

Эту функциональность достаточно легко реализовать:

1. Создайте переменную DateFilter с типом данных Text и длинной 100.



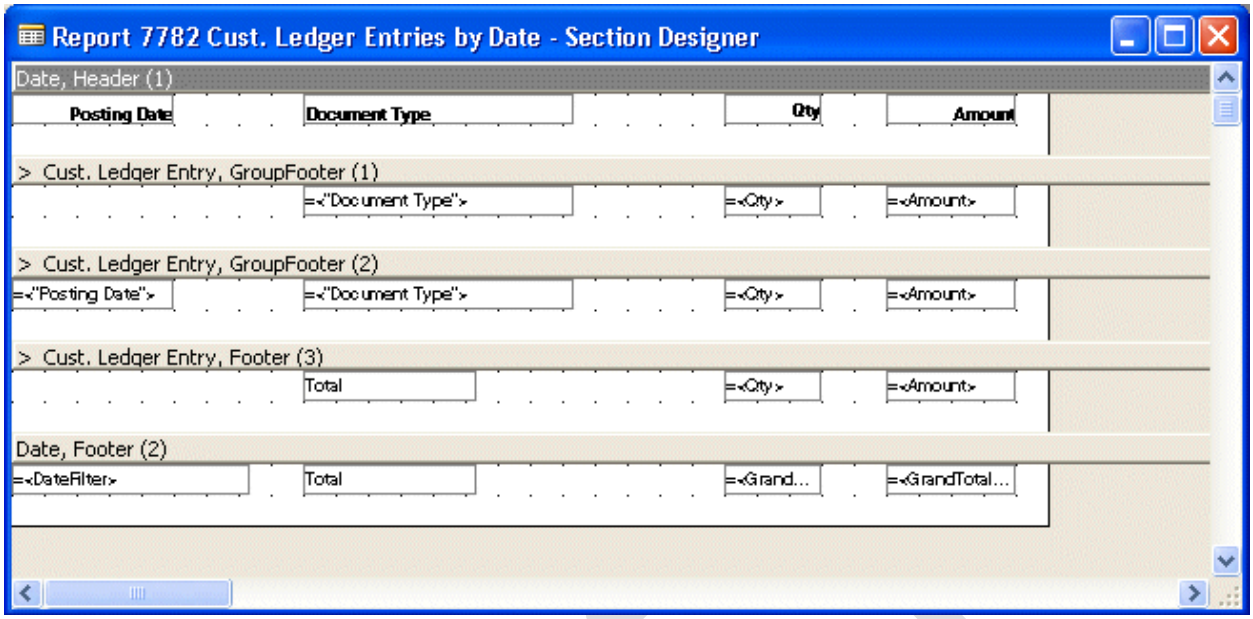
2. Добавьте следующий C/AL код в триггер *OnPreReport* отчета:

```
DateFilter := Date.GETFILTER("Period Start")
```

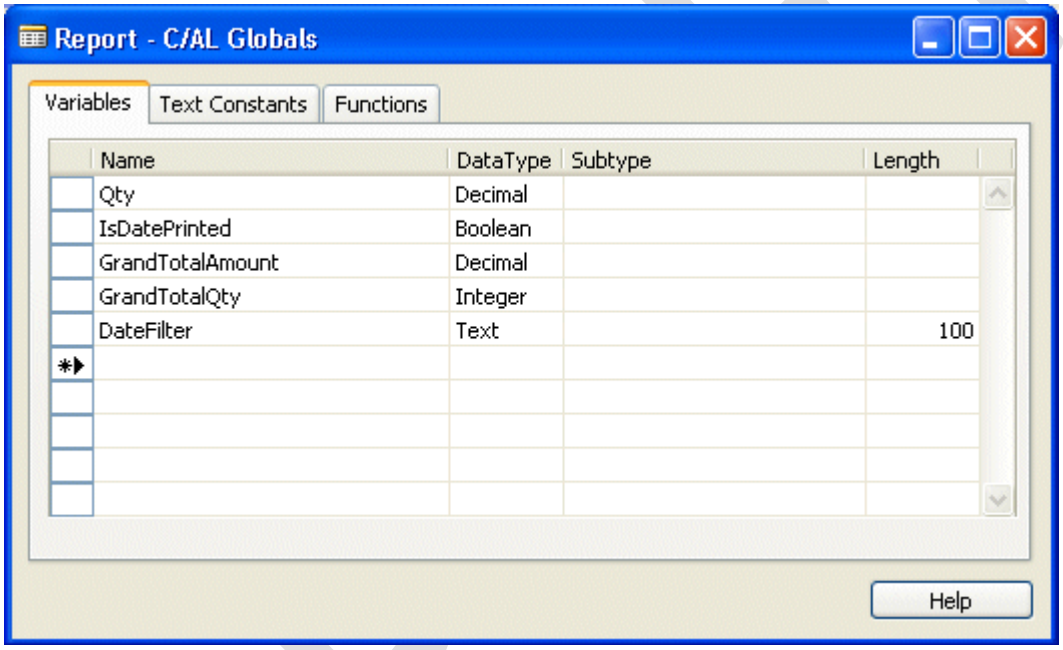
3. Добавьте текстовую зону в раздел footer элемента данных Date, откройте для нее окно ***Properties*** и введите в свойство SourceExpr значение *DateFilter*.

К тому моменту, когда система перейдет к выполнению кода в триггере *OnPreReport* форма запроса уже будет обработана. Функция GETFILTER возвращает в виде текстовой строки значения фильтров установленных для поля, переданного в качестве параметра.

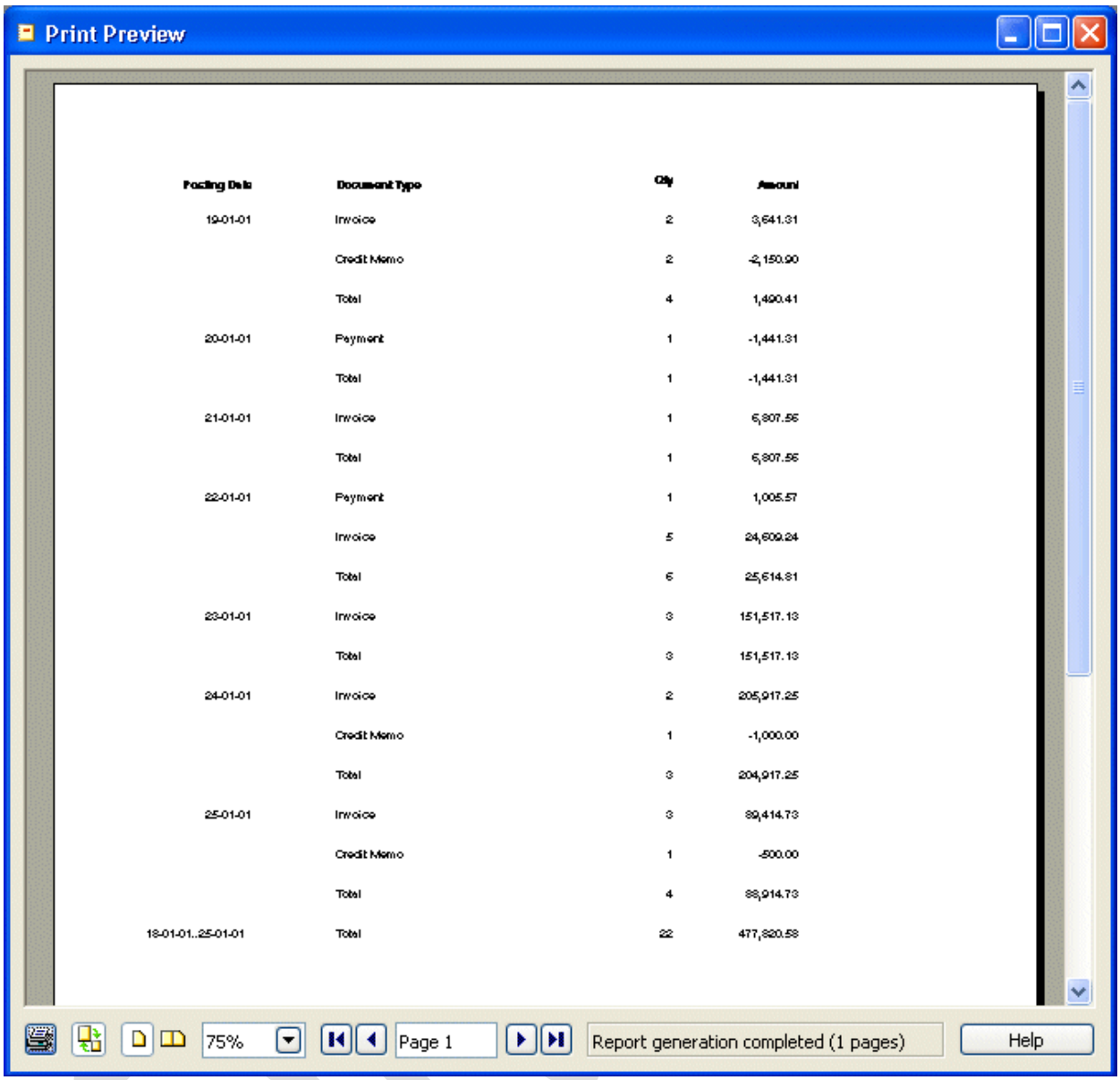
На этом создание отчета завершено, на данном этапе ***Section Designer*** должен выглядеть следующим образом:



Окно глобальных переменных ***C/AL Globals***:



После запуска отчет должен выглядеть так:



Posting Date	Document Type	Qty	Amount
19-01-01	Invoice	2	3,641.31
	Credit Memo	2	-2,150.90
	Total	4	1,490.41
20-01-01	Payment	1	-1,441.31
	Total	1	-1,441.31
21-01-01	Invoice	1	6,807.56
	Total	1	6,807.56
22-01-01	Payment	1	1,005.57
	Invoice	5	24,609.24
	Total	6	25,614.81
23-01-01	Invoice	3	151,517.13
	Total	3	151,517.13
24-01-01	Invoice	2	205,917.25
	Credit Memo	1	-1,000.00
	Total	3	204,917.25
25-01-01	Invoice	3	89,414.73
	Credit Memo	1	-500.00
	Total	4	88,914.73
18-01-01..25-01-01	Total	22	477,820.58

14.4 Создание Простого Документа

В данном разделе будет рассказано, как использовать Report Designer для создания документа (печатной формы). В примере будет создана печатная форма на основании учтенного счета продажи, в данном примере не будут приниматься в расчет особенности расчета НДС, а также не будет контролироваться ряд условий, которые должны выполняться в реальной ситуации. Кроме того, в документе не будет напечатана вся информация, которую вы привыкли видеть в счетах-фактурах.

Определение Модели Данных

Для создания счета-фактуры (sales invoice) используются две таблицы: ***Sales Invoice Header*** и ***Sales Invoice Line***. Остальные таблицы будут использоваться, чтобы детализировать коды более понятным описанием (Payment Terms, Shipment Method), а таблица ***Company Information*** будет использоваться для получения информации о компании, выставляющей счет-фактуру.

Таблица ***Sales Invoice Header*** содержит общую информацию о каждой учтенной продаже, а таблица ***Sales Invoice Line*** содержит данные о строках, входящих в документ. Таблицы связаны через поля **No**. в таблице-заголовке (это первичный ключ данной таблицы) и поле **Document No.** таблице-строках.

Чтобы определить модель данных:

1. Открыть Object Designer и создать новый пустой отчет.

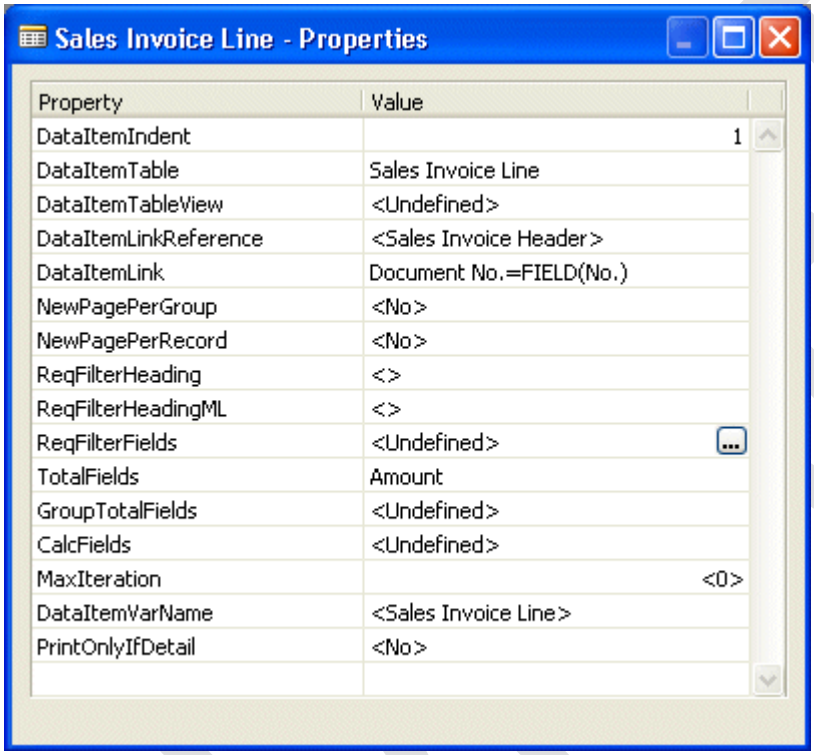
2. В окне Report Designer создать элемент данных, связанный с таблицей ***Sales Invoice Header***.

3. Создать еще один элемент данных, основанный на таблице ***Sales Invoice Line***, щелкните один раз кнопку со стрелкой вправо, чтобы задать вложенность.

4. Открыть окно ***Properties*** элемента данных Sales Invoice Line (SHIFT+F4). По умолчанию в свойстве *DataItemLinkReference* элемента данных Sales Invoice Line должна быть указана ссылка на элемент данных Sales Invoice Header. Оставьте его без изменений и укажите в свойстве *DataItemLink* значение Document No.=FIELD(No.).

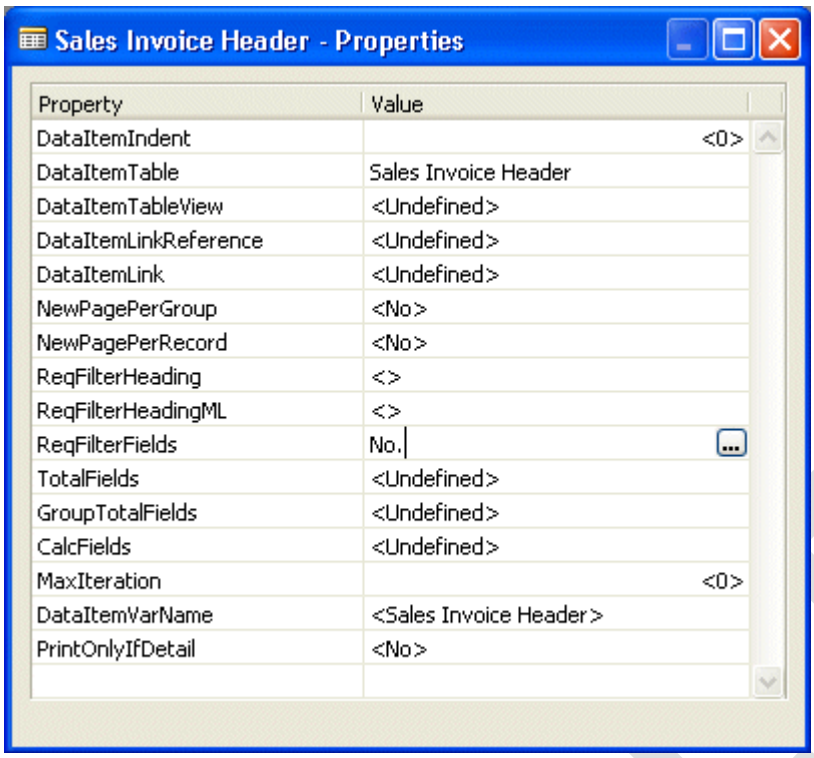
5. Укажите в свойстве *TotalFields* элемента данных Sales Invoice Line поле **Amount**. Это позволит рассчитать общую сумму по всем строкам счета-фактуры.

Окно ***Properties*** элемента данных Sales Invoice Line должно выглядеть следующим образом:



Property	Value
DataItemIndent	1
DataItemTable	Sales Invoice Line
DataItemTableView	<Undefined>
DataItemLinkReference	<Sales Invoice Header>
DataItemLink	Document No.=FIELD(No.)
NewPagePerGroup	<No>
NewPagePerRecord	<No>
ReqFilterHeading	<>
ReqFilterHeadingML	<>
ReqFilterFields	<Undefined>
TotalFields	Amount
GroupTotalFields	<Undefined>
CalcFields	<Undefined>
MaxIteration	<0>
DataItemVarName	<Sales Invoice Line>
PrintOnlyIfDetail	<No>

6. Откройте окно ***Properties*** элемента данных Sales Invoice Header. Введите в свойство *ReqFilterFields* значение **No.** Это позволит пользователям отчета, выбирать учтенный документ, для которого должна быть напечатана счет-фактура.



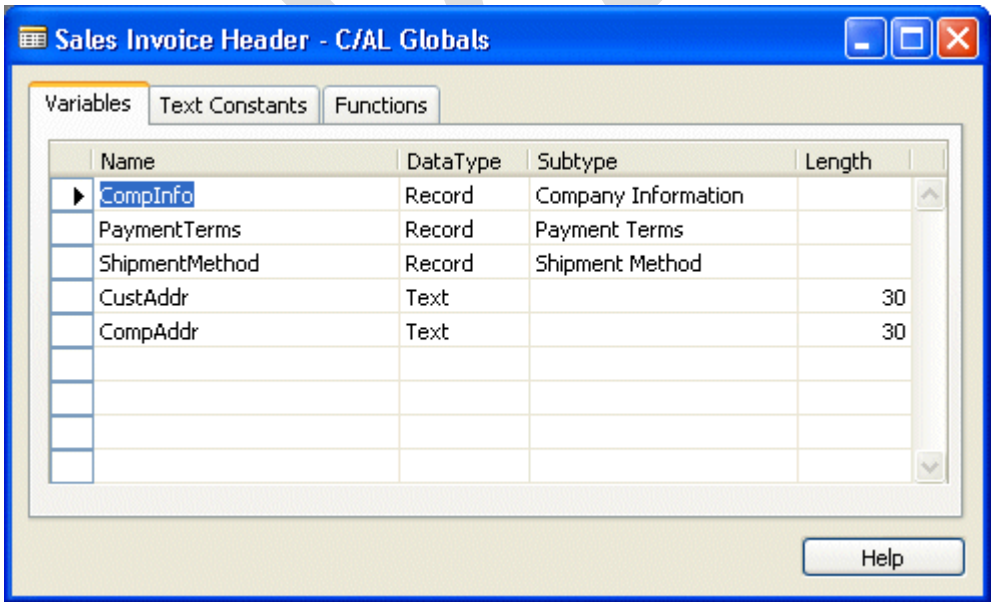
Теперь модель данных определена.

В данном отчете потребуются вспомогательные переменные для доступа и информации из других таблиц (эти таблицы не подходят для включения в модель данных).

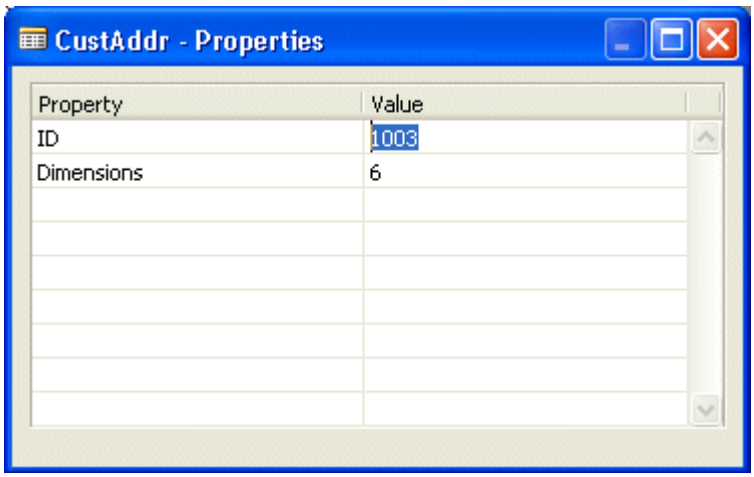
Чтобы создать переменные следует:

1. Щелкнуть Вид, C/AL Globals.

2. Объявить переменные, как показано на рисунке ниже:



3. Последние две переменные следует объявить как массивы (arrays). Откройте окно ***Properties*** для каждой из этих переменных и установите значение 6 в свойстве *Dimensions* для переменной CustAddr, и 4 для переменной CompAddr.



После того как определена модель данных, нужно ввести небольшой объем C/AL кода в триггеры отчета.

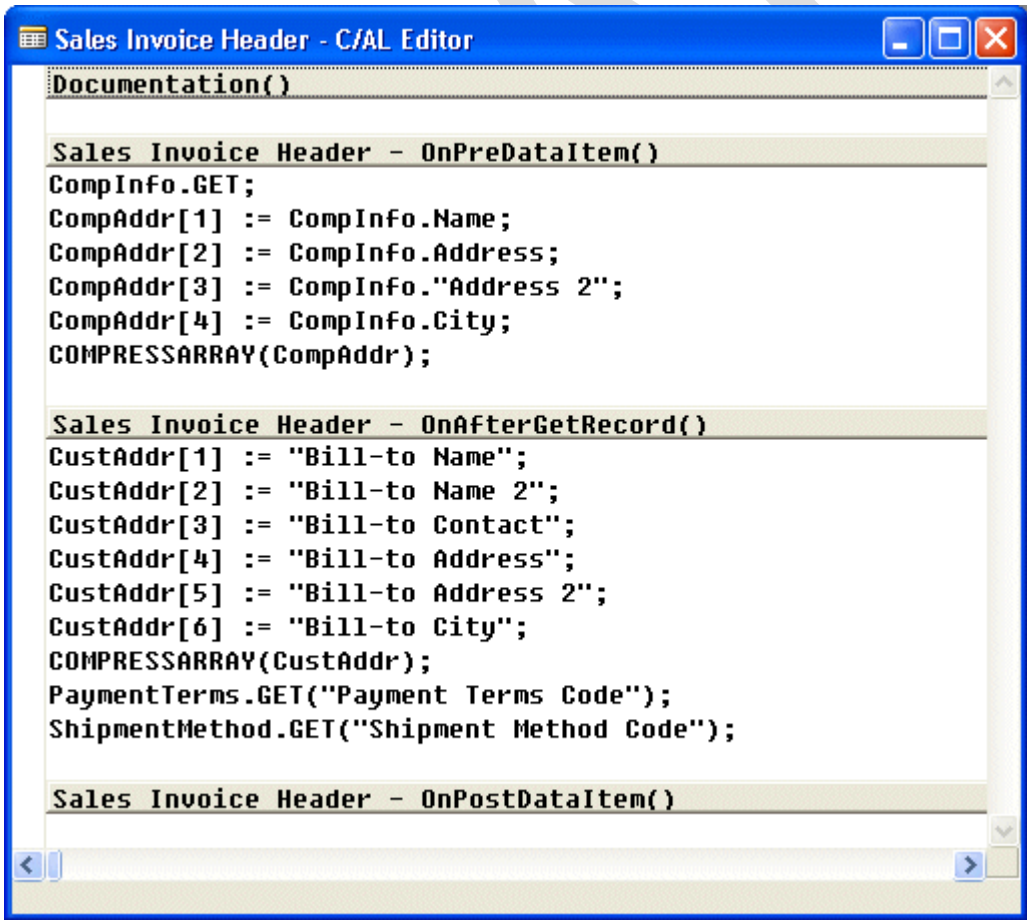
Использование Триггеров

В базовой версии отчета, для его корректной работы нужно добавить незначительный объем C/AL кода. Код должен быть добавлен в триггеры элемента данных Sales Invoice Header.

Чтобы ввести код:

1. Выберите элемент данных Sales Invoice Header и щелкните Вид, C/AL Code (F9).

На следующем рисунке приведен весь необходимый код:



Код в триггере *OnPreDataItem* работает следующим образом:

- В первой строке, CompInfo.GET, система получает единственную запись из таблицы ***Company Information***.

- В следующих четырех строках выполняется присваивание содержимого четырех полей записи таблицы ***Company Information*** элементам массива CompAddr.
- В последней строке триггера *OnPreDataItem* используется функция COMPRESSARRAY, в которую передается в качестве параметра массив CompAddr. Эта функция исключит из массива пустые строки. Это нужно сделать, так как поля таблицы могли содержать пустые значения. Если просто вывести на печать каждый элемент массива на отдельной строке, то могут быть напечатаны пустые строки.

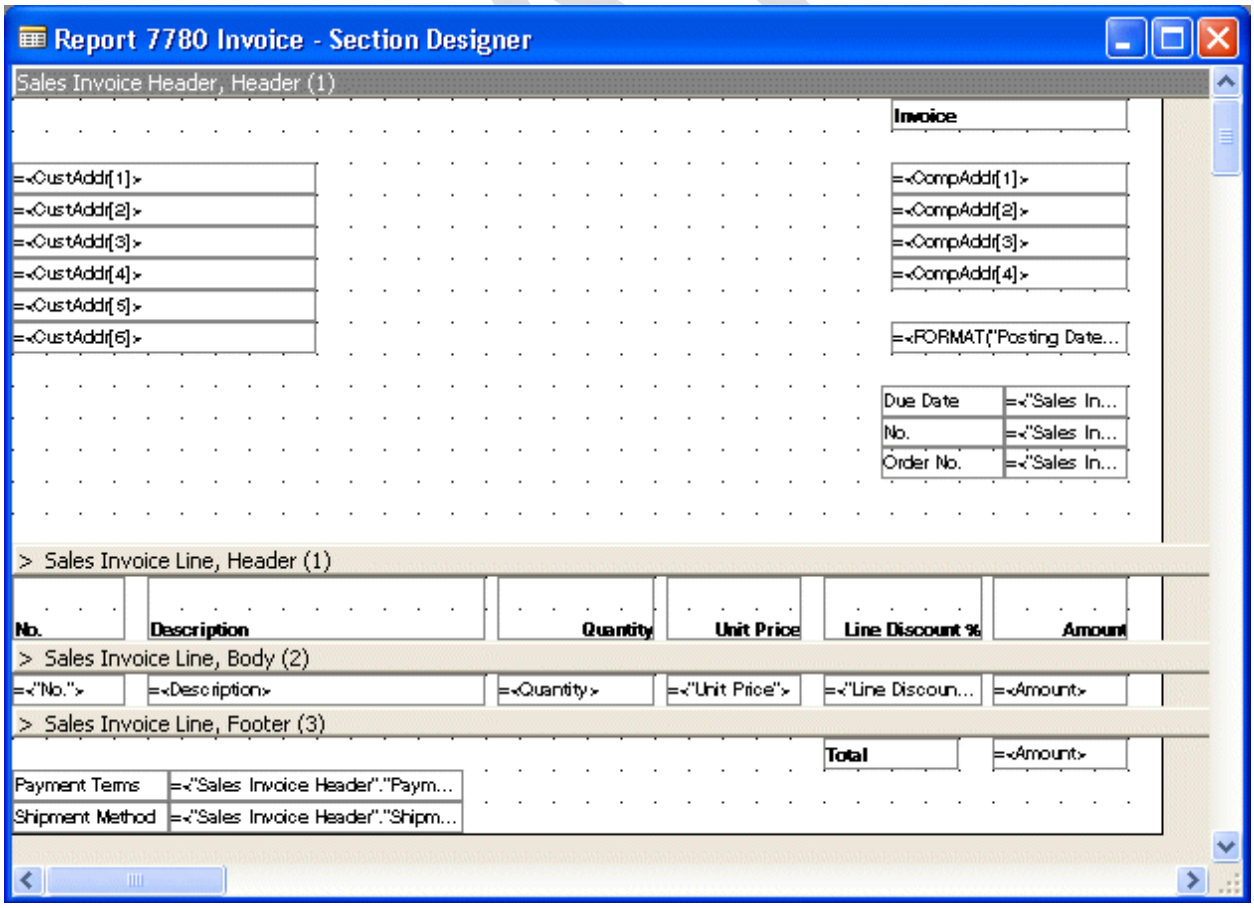
Код в триггере *OnAfterGetRecord* работает следующим образом:

- В первых шести строках выполняется присваивание содержимого шести полей записи элемента данных Sales Invoice Header элементам массива CustAddr.
- Далее вызывается функция COMPRESSARRAY, чтобы исключить из массива пустые строки.
- В последних двух строках используется функция GET (со значениями полей **Payment Terms** и **Shipment Method** из записи Sales Invoice Header в качестве передаваемых параметров) для получения соответствующих записей из таблиц ***Payment Terms*** и ***Shipment Method***. При создании разделов, полнотекстовые описания могут быть получены из данных записей.

Создание Разделов

После того как была определена модель данных и написан код на языке C/AL, использующийся для получения дополнительной информации, можно перейти к созданию секций.

На следующем рисунке показан Section Designer, после того как были вставлены необходимые разделы и добавлены соответствующие элементы управления:



В разделе Header элемента данных Sales Invoice Header нужно обратить внимание на следующие моменты:

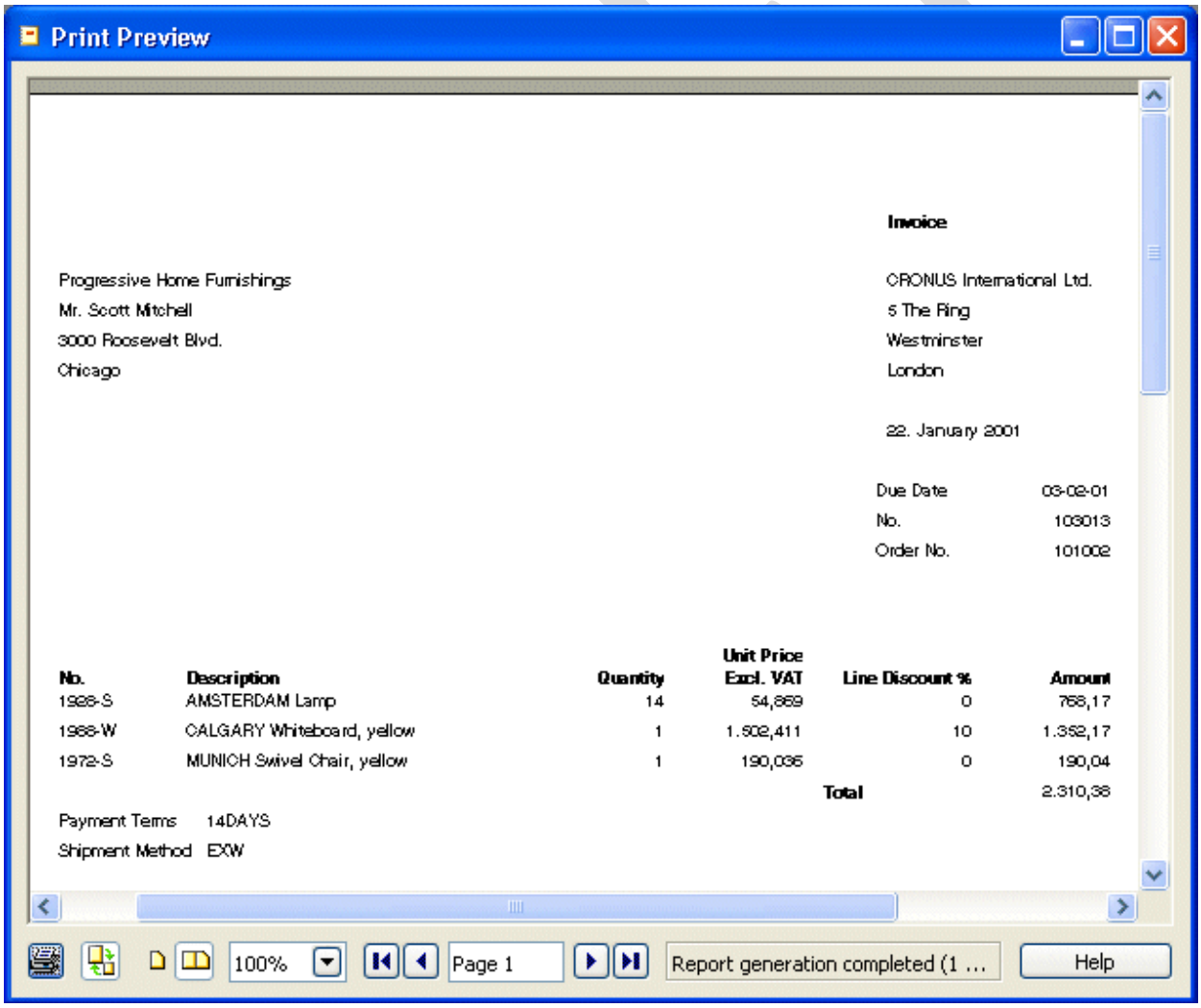
- Было добавлено шесть текстовых зон со значениями CustAddr[1]..CustAddr[6] в SourceExpr. Если вы сравните раздел с тем документом, который показан ниже, то заметите, что в документе выведено только четыре элемента содержащих данные. Благодаря использованию функции COMPRESSARRAY пустые строки были удалены, а данные собраны вместе.
- Аналогично, в документе, показанном ниже, данные отображаются только в трех элементах массива CompAddr.
- В текстовой зоне, в которой выводится дата учета, в качестве выражения используется не **Posting Date**, а выражение на языке C/AL: FORMAT("Posting Date",0,4). Оно используется для изменения форматирования даты.
- В разделе Footer элемента данных Sales Invoice Line выводится поле **Amount**, содержащее итоговую сумму по всем строкам, напечатанным в разделах Body.
- В том же разделе выводится полнотекстовое описание полей Payment Terms и Shipment Method, для этого в качестве SourceExpr используются:

  "Sales Invoice Header"."Payment Terms Code"

  и

  "Sales Invoice Header"."Shipment Method.Code"

На следующем рисунке показано, как должна выглядеть счет-фактура:



14.5 Создание Непечатаемого Отчета

Последним будет создан непечатаемый отчет. Хотя аналогичную функциональность можно получить, написав программный модуль, существует ряд причин, по которым следует использоваться непечатаемые отчеты:

- Доступна функциональность форм запроса (можно ввести параметры и установить фильтры), чтобы повторить аналогичную функциональность в программном модуле потребуется значительный объем программирования на языке C/AL.
- Использование функциональности Report Designer для ввода параметров и установки фильтров обеспечивает единообразие создаваемого приложения.
- Вместо написания C/AL кода по открытию таблиц и получению из них записей, достаточно определить элементы данных.

Данный простой отчет используется для корректировки цен в таблице Item (Товары). Пользователи могут устанавливать фильтры на поля таблицы, ограничивая список обновляемых товаров по номерам, учетным группам или поставщикам, также они могут задать параметры, влияющие на изменение цены.

Определение Модели Данных

Данный отчет состоит из одного элемента данных, который основывается на таблице ***Item***.

Чтобы определить модель данных:

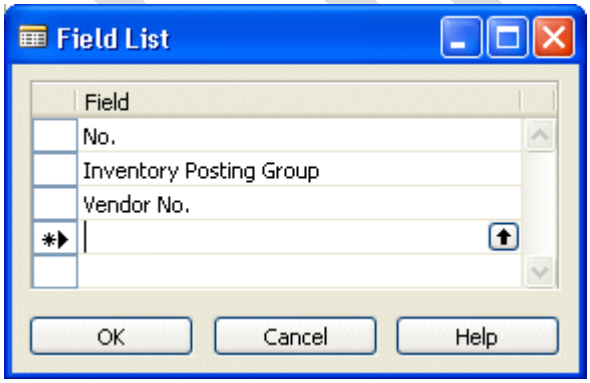
1. Откройте Report Designer и создайте новый пустой отчет.

2. Создайте элемент данных на основании таблицы ***Item***.

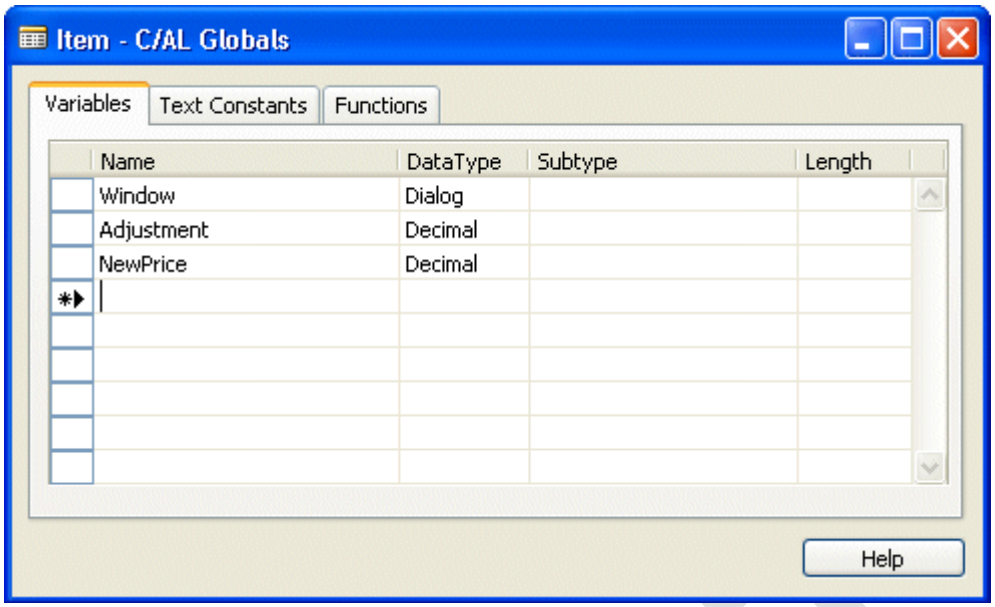
3. Откройте окно ***Properties*** для отчета и установите в свойстве *ProcessingOnly* значение Да.

4. Откройте окно ***Properties*** для элемента данных и установите в свойстве *DataItemTableView* значение ключа No.. Хотя это не обязательно для работы, в результате из формы запроса исчезнет кнопка Сортировка. Так как отчет не будет выводиться на печать, то порядок, в котором будет выполняться обработка записей, не важен.

5. Щелкните кнопку AssistButton в свойстве *ReqFilterFields*, чтобы указать поля, по которым пользователи смогут устанавливать фильтры:



6. Объявите следующие три переменные:



- Переменная Window с типом данных Dialog будет использоваться для отображения информации на экране.
- Переменная Adjustment используется для ввода значений на форме запроса.
- Переменная NewPrice используется для хранения промежуточных результатов.

Создание Формы Запроса

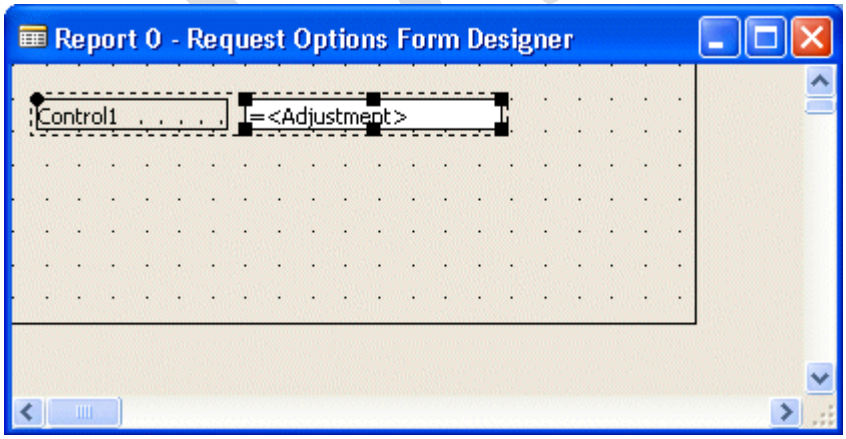
В предыдущих примерах, создаваемые формы запроса содержали закладки с фильтрами для соответствующих элементов данных. В данном случае нужно добавить закладку Параметры, где можно будет ввести параметр коррекции.

Чтобы создать закладку **Параметры**:

1. Щелкните Вид, Request Form, чтобы открыть окно **Request Options Form Designer**.

2. Добавьте текстовую зону с надписью (чтобы надпись добавилась автоматически при вставке текстовой зоны, щелкните кнопку Add Label на панели инструментов Toolbox, перед тем как вставлять текстовую зону на форму).

3. В окне ***Properties*** текстовой зоны укажите для свойства *SourceExpr* значение Adjustment (переменная созданная ранее):

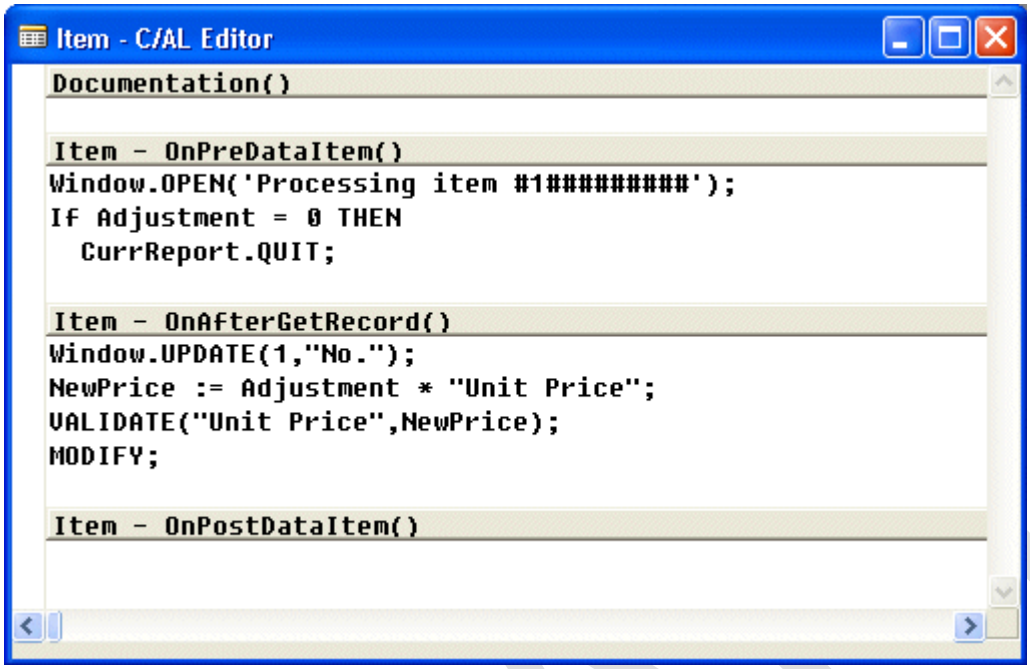


DRAFT

Использование Триггеров

После того как было определена модель данных и создана форма запроса нужно добавить в триггеры элемента данных Item некоторое количество C/AL кода, в котором будет выполняться обновление цены.

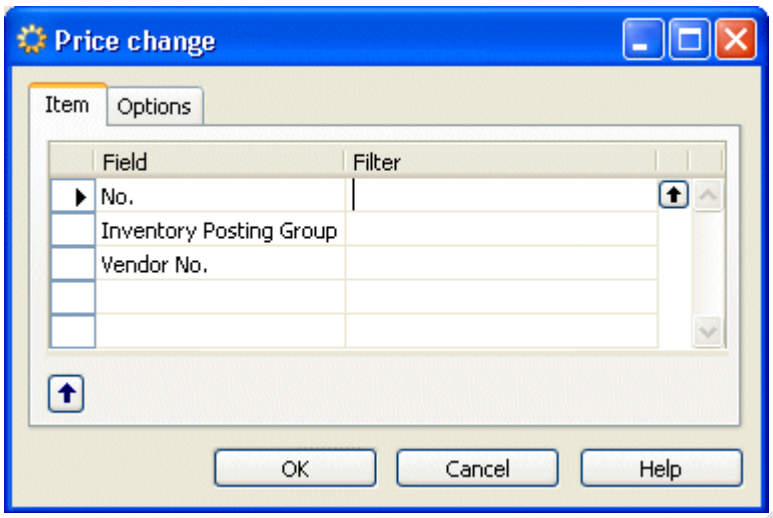
На следующем рисунке показан необходимый C/AL код:



Приведенный код работает следующим образом:

- Первое выражение в триггере *OnPreDataItem* открывает окно, в котором будет отображаться ход выполнения отчета. Так как отчет не будет печататься, то обычное окно статуса печати не будет отображаться. Если таблица содержит много записей, процесс обработки может занять некоторое время. Поэтому удачной идеей будет предоставить пользователю информацию о ходе выполнения.
- В первой строке триггера *OnAfterGetRecord* в окно выводится номер товара, каждый раз при получении новой записи из базы данных.
- Второе выражение в триггере *OnPreDataItem* завершит отчет, не выполняя никаких изменений, если параметр коррекции равен 0 (ноль). Если не выполнить данную проверку, то все цены в таблице станут равны нулю. В более дружественной версии отчета можно предложить пользователю ввести значение параметра коррекции еще раз или, по крайней мере, сообщить о причинах отмены выполнения отчета.
- Последние три строки триггера *OnAfterGetRecord* выполняют обновление цен. В первой строке рассчитывается скорректированная цена и присваивается переменной NewPrice. Затем с помощью функции VALIDATE выполняется обновления значения поля **Unit Price**. При этом выполняются специальные процедуры (например, обновления связанных полей), заложенные в триггере *OnValidate* поля таблицы. И наконец, функция MODIFY выполняет запись изменений в базу данных.

При запуске отчета, форма запроса будет выглядеть следующим образом:



На первой закладке установите фильтры по номерам товаров, учетным группам и поставщикам.

На закладке Параметры, укажите значение, на которое должна быть скорректирована цена.

14.6 Типы Отчетов

В этом разделе вы узнаете о типах отчетов, которые обычно находятся в функциональной области. Типы отчетов не так строго определены, как типы форм или таблиц. Это лишь краткая схема, которая поможет вам в дальнейшем при изучении архитектуры системы.

Отчеты List

Отчет List содержит единственный элемент данных, связанный с таблицей, для которой будет сформирован список. Это либо таблица Master, либо таблица Supplemental.

Каждый столбец содержит поле из таблицы, и данные печатаются из этой таблицы, а не передаются из других таблиц и не вычисляются на основе данных из других таблиц.

Именем обычно является имя таблицы, за которым следует слово «List»

Примеры:

- Customer - List
- Insurance - List
- Vendor - List

Отчеты Test

Отчет Test – это отчет, выведенный на печать из таблицы Journal. Его назначение состоит в том, чтобы проверить каждую строку таблицы Journal Lines согласно тем критериям, которые будут использоваться для учета с тем, чтобы можно было обнаружить и исправить все ошибки перед учетом.

Это полезно, так как при обнаружении ошибки во время учета обработка останавливается, и ошибка должна быть исправлена перед продолжением учета.

Таким образом, использование отчета Test перед учетом таблицы Journal, содержащей много ошибок, хороший способ обнаружения этих ошибок.

Именем обычно становится имя соответствующей формы Journal, за которым следует слово «Test».

Примеры:

- General Journal - Test
- Resource Journal - Test

DRAFT

Отчеты Posting

Отчет Posting можно печатать как часть опции «Учет и Печать» в журнале. Фактически это отчет, который выводится на печать из окна регистр, и он имеет то же имя, что и соответствующая таблица Register.

В нем перечислены все операции (то есть Ledger Entries), которые были зарегистрированы в этом регистре.

Пример:

- G/L Register
- Vendor Register

Отчеты Transaction

Отчет Transaction содержит два элемента данных. Первый – это таблица Master, второй - соответствующая таблица – книга операций (Ledger).

Обычно отчет Transaction содержит список всех записей в таблице Ledger связанных с таблицей Master

Как правило, дается промежуточная сумма для каждой записи таблицы Master и итоговая сумма для всех напечатанных таблиц.

Это отчет используется для просмотра всех сделок для определенной записи Master.

У этого типа отчетов нет стандартного имени.

Примеры:

- Trial Balance
- Vendor - Trial Balance

Другие "Нормальные" Отчеты

Отчеты определяются более расплывчато, чем другие объекты приложения, потому что их часто настраивают под конкретного клиента. Тем не менее, большинство отчетов состоит из табличной распечатки с записями, расположенными горизонтально, и каждое поле отображено в собственном столбце. Часто вводится некий заголовок группы или общее количество, что позволяет разбить строки на разные категории и найти промежуточные суммы по этим категориям.

Примеры:

- Vendor Information
- Item Sales by Customer

Отчеты Document

Отчеты Document отличаются от большинства других отчетов тем, что многие из полей не отображаются в столбцах.

Примером этого типа отчетов может послужить счет-фактура (Invoice). В данном отчете информация из заголовка повторяется наверху каждой страницы, для каждой записи таблицы-заголовка, информация выводится на отдельной странице. Строки для счета-фактуры выводятся на печать как обычный отчет, состоящий из строк и столбцов.

Пример:

- Sales Invoice Nos
- Purchase Invoice Nos

DRAFT

ЧАСТЬ 6 ПРОГРАММНЫЕ МОДУЛИ

ГЛАВА 15 ПРОГРАММНЫЕ МОДУЛИ – БАЗОВЫЕ ПОНЯТИЯ

В этой главе дается объяснение того, что такое программный модуль и как его создавать. В ней также показано, как функции программного модуля вызывать из других прикладных объектов.

- Что такое программный модуль среды C/SIDE?
- Создание программных модулей
- Использование программных модулей

15.1 Что такое Программный Модуль Среды C/SIDE?

В предыдущих частях данной книги уже встречались примеры использования C/AL кода на формах и отчетах. Фактически C/AL-код можно сохранять в форме, отчете или другом объекте. Обычный подход при работе с простыми приложениями заключается в том, чтобы помещать код в объект, вызывающий функции. Но по мере разрастания своего приложения вы будете обнаруживать, что часто используете одинаковые функции. Вместо того чтобы снова и снова объявлять одни и те же функции, вы могли бы определить их единожды. В этом случае полезно использовать программный модуль (codeunit). Представьте себе программный модуль в виде контейнера для кода C/AL, который вы хотите использовать во многих прикладных объектах.

В программных модулях вы можете определить следующее:

Функции

Функция (function) – это последовательность выражений на языке C/AL, которую вы определяете с целью создания новых функциональных возможностей. В каждой функции можно определить локальные переменные, то есть переменные, область действия которых ограничена той функцией, где они определены.

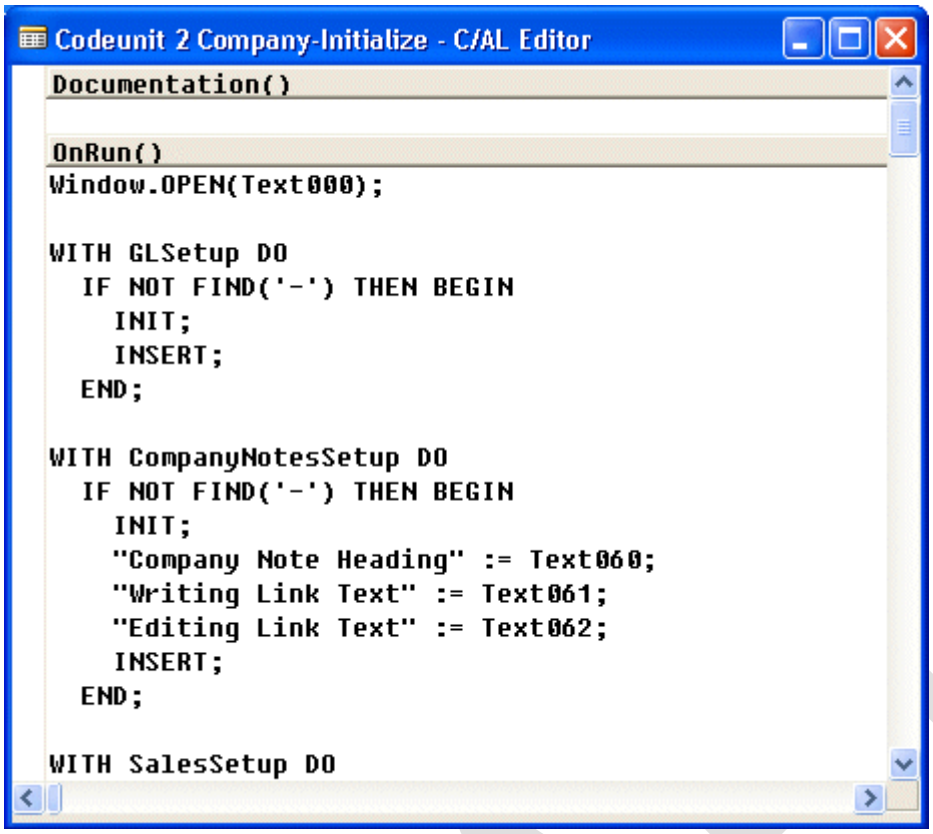
Глобальные переменные

Глобальная переменная (global variable) – это переменная, область действия которой охватывает все функции в программном модуле.

Временные таблицы

Временная таблица (temporary table) – это таблица, которая не сохраняется в базе данных. Временные таблицы используются в основном как структурированные переменные, в которых временно хранятся данные, с которыми вы работаете. Обратитесь к разделу «Определение и использование временной таблицы», где дано описание того, как создаются временные таблицы.

DRAFT



Все программные модули содержат два триггера по умолчанию, называемые Documentation и OnRun. В секцию Documentation вы можете добавлять необязательную информацию о коде, например, о назначении программного модуля, о номере версии и т.д. В секцию OnRun вы можете включить код, который должен исполняться системой при запуске программного модуля.

> **Примечание**
>
> Программные модули содержат функции, но тоже могут исполняться. Будучи контейнером для функций, которые можно запускать по отдельности, программный модуль сам может исполняться при условии написания кода <CodeunitVariableName>.Run. Запуская программный модуль, вы исполняете код в секции OnRun программного модуля.

15.2 Создание Программных Модулей

Для создания нового программного модуля или изменения существующего используется Object Designer.

Чтобы создать программный модуль:

1. Откройте **Object Designer** (SHIFT+F12) и щелкните кнопку **Codeunit**.

2. Щелкните кнопку **New**, чтобы создать новый программный модуль. Откроется окно C/AL Editor, в котором можно создавать требуемые функции.

Чтобы изменить существующий программный модуль:

1. Откройте **Object Designer** (SHIFT+F12) и щелкните кнопку **Codeunit**

2. Выберите программный модуль, который требуется модифицировать, и щелкните кнопку **Design**. Откроется окно C/AL Editor, в котором можно изменять существующие функции или добавлять новые.

Использование C/AL Editor

C/AL Editor предназначен для облегчения задачи создания и модификации C/AL кода. При работе с C/AL Editor, вы получаете доступ к C/AL Symbol Menu, которое помогает определять функции C/AL.

DRAFT

Используя C/AL Symbol Menu, можно получить справку обо всех функциях C/AL. Выберите имя функции C/AL в правом столбце и нажмите клавишу F1. Больше информации о C/AL Symbols Menu см. в Использование C/AL Symbol Menu.

В процессе создания программного модуля в окне отображаются две определенные по умолчанию секции, описанные выше (секции Documentation и OnRun).

Из окна Дизайнера объектов вы можете открывать необходимое количество программных модулей. Каждый создаваемый вами или открываемый программный модуль будет отображаться в отдельном окне. Это облегчает задачу вырезания строк кода из одного программного модуля и их вставки в другой модуль.

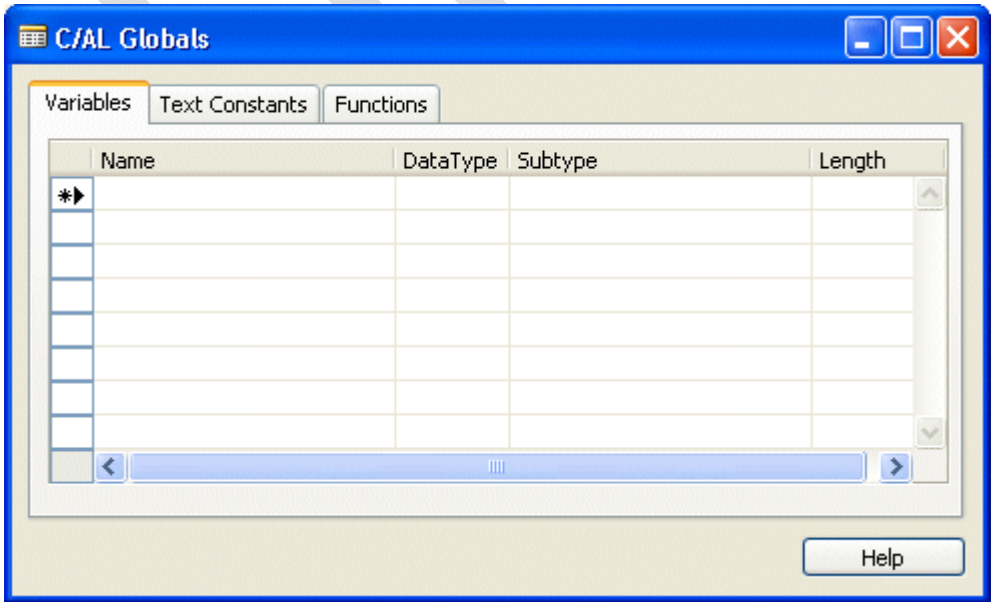
Если у вас есть опыт использования других редакторов в среде Windows, вы поймете, насколько Редактор C/AL-кода прост в использовании. Вы можете получить доступ к функциям редактирования как из меню Правка, так и из панели инструментов. Также доступны стандартные горячие клавиши:

Чтобы...	Нажмите...
Вырезать выделенный текст и поместить его в буфер обмена	CTRL+X
Скопировать выделенный текст в буфер обмена	CTRL+C
Вставить текст из буфера обмена в программный модуль в точку нахождения курсора	CTRL+V
Открыть окно ***Найти*** для поиска названий триггеров	CTRL+F

Определение Переменных, Текстовых Констант и Функций в Программном Модуле

Создав программный модуль, вы должны сделать следующий шаг – определить глобальные переменные, текстовые константы и функции для этого программного модуля. Для этого используется окно C/AL Globals.

Чтобы получить доступ к C/AL Globals, убедитесь в том, что выбран C/AL Editor, щелкните Вид, C/AL Globals. Среда C/SIDE отобразит окно **C/AL Globals**.



В окне ***C/AL Globals*** можно добавлять глобальные переменные, текстовые константы или функции.

Переменные. Чтобы добавить глобальную переменную:

1. Перейдите на закладку ***Variables*** в окне ***C/AL Globals***.

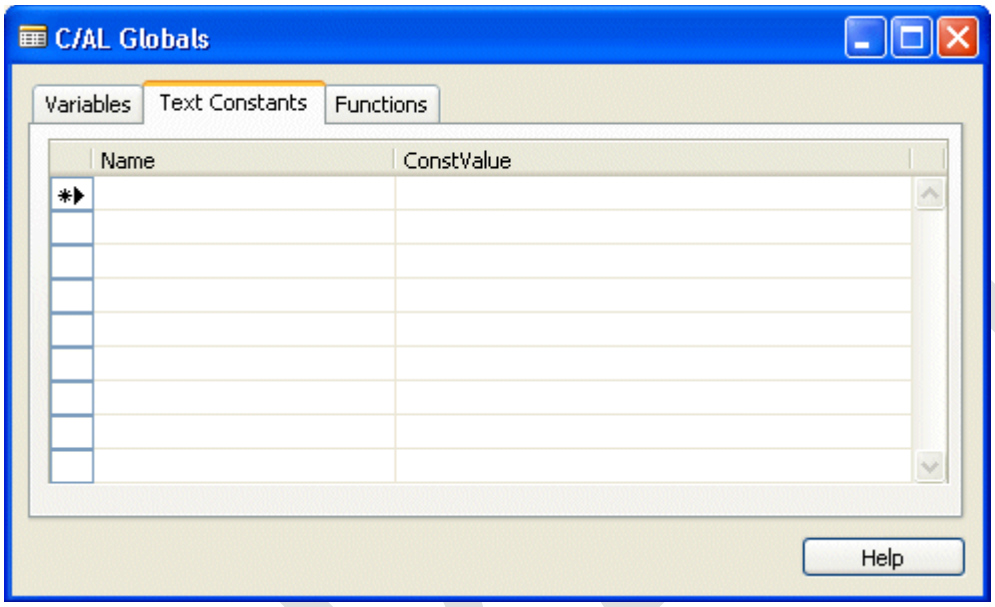
2. Введите имя переменной, укажите ее тип. Если выбранный тип связан с объектом приложения, то также можно указать подтип – наименование конкретного объекта базы данных.

Если выбран тип данных text или code, необходимо определить длину переменной (по умолчанию длина переменной типа code составляет 10, а text - 30).

Если выбран тип данных OCX или Automation также требуется указать подтип. См. ГЛАВА 19 РАСШИРЕНИЕ ВОЗМОЖНОСТЕЙ ЯЗЫКА C/AL.

Текстовые константы. Чтобы создать текстовую константу, которая будет использоваться в качестве сообщения для пользователя нужно:

1. Перейдите на закладку ***Text Constants*** в окне ***C/AL Globals***:



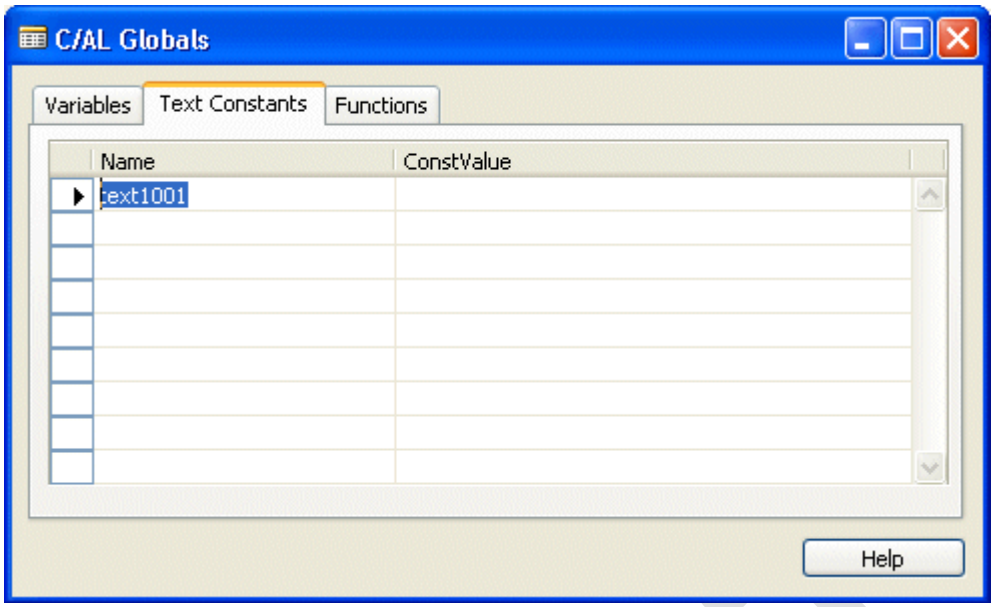
2. В первом поле **Name** ввести наименование новой текстовой константы.

> **Примечание**
>
> Для наименований текстовых констант правила не предусмотрены. Использование уникального идентификатора в наименовании желательно, но обязательно.

3. Откройте окно **Properties** для текстовой константы.

Для текстовой константы автоматически был присвоен уникальный идентификатор (поле **ID)**.

4. Скопируйте идентификатор в поле **Name** окна ***C/AL Globals***, например, для ID = 1001 должно получиться наименование *Text1001*.



5. Щелкните кнопку AssistButton в поле **ConstValue,** чтобы открыть окно ***Multilanguage Editor***.

6. В поле **Language** введите ENU для английского языка или RUS для русского.

7. В поле **Value** введите текстовую строку, которую должна содержать текстовая константа.

8. Щелкните кнопку OK, чтобы выйти. Если не нажать кнопку OK информация не сохранится.

9. В C/AL Editor поместите наименование текстовой константы в функцию, отображающую сообщение или ошибку.

Пример

```
IF FileName = '' THEN
  ERROR(Text1001);
```

При установке курсора в окне C/AL Editor на название текстовой константы ее содержимое отображается в строке состояния.

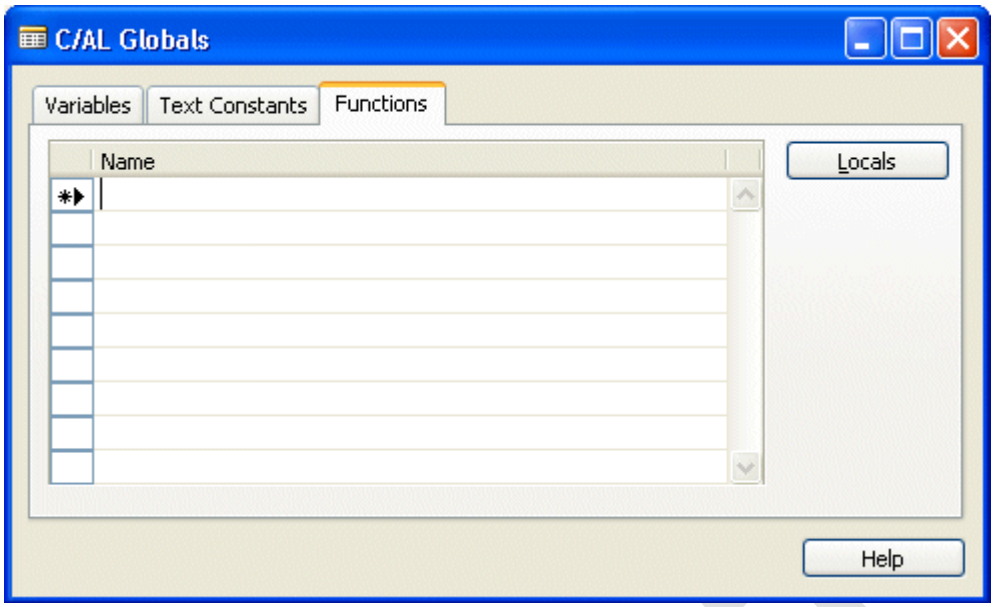
Примечание

Если ввести текстовую строку непосредственно в поле **ConstValue** окна ***C/AL Globals***, а затем открыть Multilanguage Editor, то можно увидеть, что текст сохранен в соответствии с текущим выбранным языком приложения.

Функции. Чтобы добавить функцию:

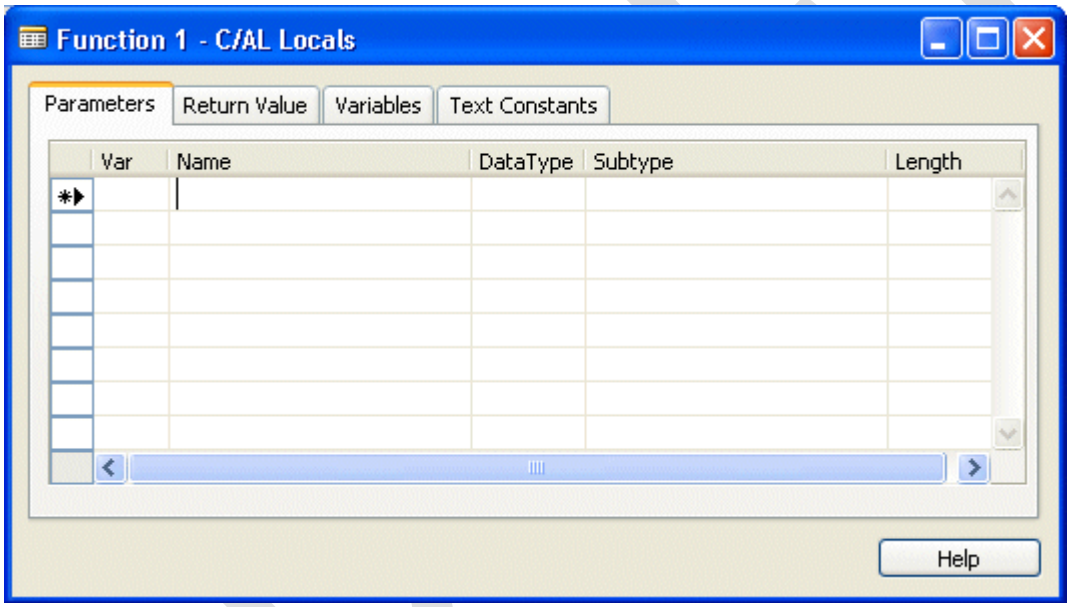
1. Перейдите на закладку ***Functions*** окна ***C/AL Globals***:

DRAFT



2. Введите название каждой новой функции.

3. Щелкните кнопку Locals, чтобы определить передаваемые параметры, возвращаемое значение, локальные переменные и текстовые константы для каждой функции. Откроется окно ***C/AL Locals***:

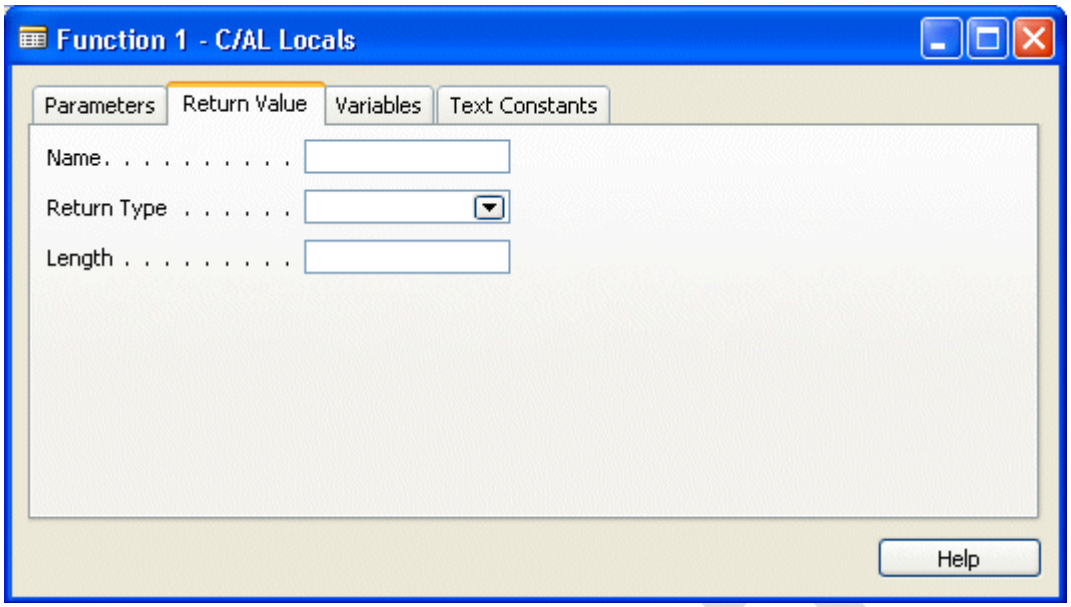


4. Для каждого параметра вы должны указать метод вызова, имя и тип данных. Вы можете определить подтип и длину, но это необязательно.

Метод вызова можно определить как Var – это означает, что параметр передается по ссылке, а не по значению. Значение переменной может изменяться функцией только в том случае, когда она передается в функцию по ссылке. Если параметр не определен как Var, функции передается только копия переменной. Если функция изменит значение, это изменение затронет копию, а не саму переменную.

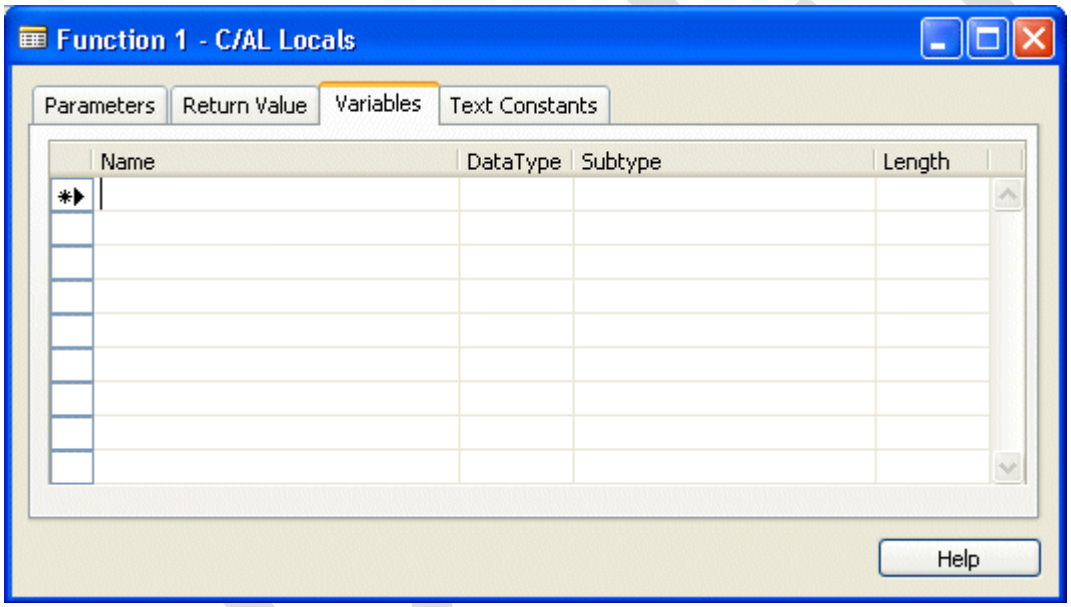
Если выбранный вами тип относится к прикладному объекту, вы также должны добавить подтип, т.е. имя конкретного объекта в БД. Если вы выбираете текст или код, необходимо определить его длину (длина кода по умолчанию составляет 10 символов, длина текста – 30 символов).

5. Перейдите на закладку ***Return Value*** возвращаемое значение для новой функции.



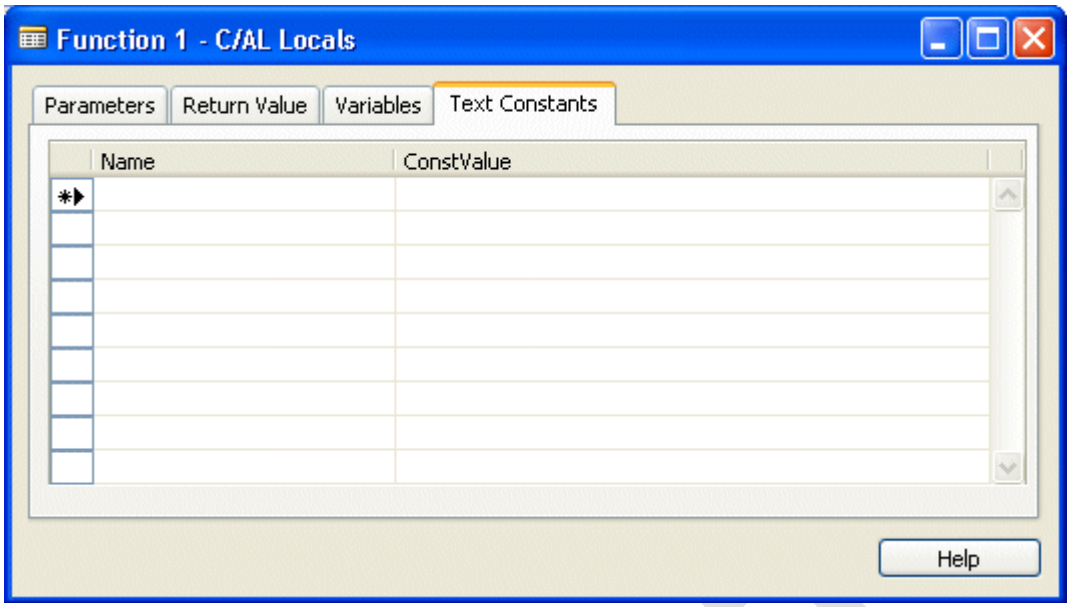
6. Введите имя возвращаемого значения и выберите тип данных из раскрывающегося списка в поле **Return Type**. Кроме того, вы можете выбрать длину, но только если типом является текст или код.

7. Перейдите на закладку ***Variables,*** чтобы определить локальные переменные для функции:



8. Вы должны добавить имя и тип для каждой локальной переменной. Если выбранный вами тип относится к прикладному объекту, вы также должны добавить подтип, т.е. имя конкретного объекта в БД. Если вы выбираете текст или код, необходимо определить длину переменной (длина кода по умолчанию составляет 10 символов, длина текста – 30 символов).

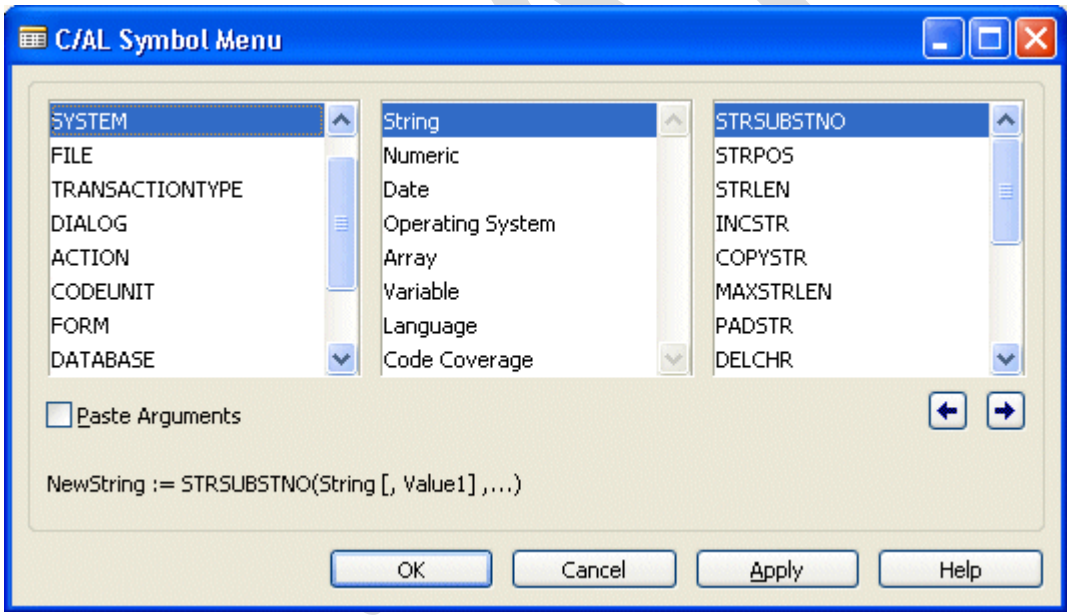
9. Перейдите на закладку ***Text Constants,*** чтобы определить текстовые константы для функции.



Использование C/AL Symbol Menu

При написании кода C/AL в редакторе вы можете использовать меню ***/AL Symbol Menu***, чтобы иметь возможность просмотра:

- всех переменных, определенных в программном модуле;
- всех функций C/AL.



Окно ***C/AL Symbol Menu*** разделено на три столбца:

- В левом столбце показаны имена переменных (если они определены) и категории функций.
- Во втором столбце отображаются подкатегории.
- В третьем столбце отображаются функции выбранной категории.

Синтаксис и прочая информация, такая как свойство *Caption* выбранного поля, отображается в нижней части окна.

В ряде случаев, например, для элементов форм или сопутствующих форм, а также для BLOB-полей, используется больше трех колонок.

Щелкните кнопку OK или Apply в окне **C/AL Symbol Menu**, чтобы вставить выбранную функцию или переменную в C/AL Editor. При нажатии на кнопку OK окно ***C/AL Symbol Menu*** автоматически закрывается, при нажатии на кнопку Apply окно остается открытым.

Если вам нужна справочная информация о какой-либо функции C/AL, отображаемой в столбце справа, выберите имя функции и нажмите клавишу F1, чтобы активизировать контекстно-зависимую онлайновую справку *C/SIDE Reference Guide*.

Компиляция и Сохранение Программного Модуля

Перед выполнением функций, находящихся в программном модуле, необходимо откомпилировать и сохранить программный код. При компиляции кода система проверяет синтаксис выражений. Если компилятор обнаруживает ошибки в коде, он выведет на экран сообщение.

Чтобы откомпилировать код в программном модуле, выполните следующие действия.

1. Щелкните Сервис, Compile.

2. Если система обнаружит в коде ошибки, она выведет на экран сообщение. Исправьте ошибку и снова запустите компиляцию.

Чтобы сохранить программный модуль:

1. Щелкните Файл, Save, среда C/SIDE откроет окно ***Save As***.

2. Введите уникальный идентификатор и наименование объекта.

3. Укажите должна ли система при сохранении выполнять компиляцию.

Если вы сохраните программный модуль, не компилируя его, вы не сможете исполнять или вызывать его функции.

> **Примечание**
>
> Зачем сохранять код, не выполнив компиляцию? При работе над большим и сложным программным модулем вам необходимо регулярно сохранять свою работу, даже если она еще не закончена и не может быть откомпилирована. В этом случае вы должны снять флажок из окна Compiled, прежде чем производить сохранение.

15.3 Использование Программных Модулей

Используя программные модули, вы устраняете необходимость дублирования кода и в то же время облегчаете поддержку кода. Если вы часто используете свой код в формах или отчетах, необходимо создать функцию в программном модуле. Создав функцию в программном модуле, вы можете получить к ней доступ с помощью следующего кода:

<CodeunitName>.<FunctionName>

> **Пример**
>
> Предположим, что вы создали программный модуль, содержащий две статистических функции с именами F и G. На листинге, приведенном ниже, показано, как получить доступ к этим функциям из формы

Программный модуль с именем named StatisiticFunctions

```
F(x:integer)
Begin
...
End
```

DRAFT

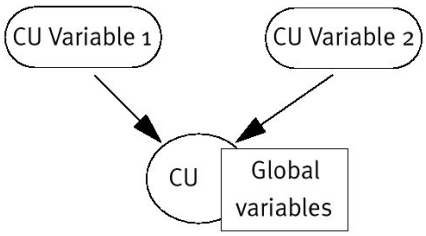
```
G(x:integer)
Begin
...
End
```

Любая форма

```
//глобальная переменная StatFun объявлена как Codeunit с подтипом StatisticsFunctions
...
Result :=
StatFun.F(3425)+StatFun.G(346);
...
```

Этот метод является общеприменимым. То есть вы можете получить доступ из любого прикладного объекта к функциям в других прикладных объектах, поставив перед именем функции имя прикладного объекта, содержащего функцию.

Вы можете получать доступ к программным модулям через переменные программного модуля – либо явно объявив переменную с программным модулем типа данных, либо установив свойство RunObject в формах на программный модуль. Переменная программного модуля содержит не программный модуль, а лишь ссылку на него. На один и тот же программный модуль могут ссылаться несколько переменных программного модуля. Это проиллюстрировано на следующем рисунке.



Программные модули содержат внутренние переменные, которые определяются как глобальные переменные. К этим переменным нельзя получить прямой доступ из кода, находящегося вне данного программного модуля, но к ним можно получить доступ через определенные пользователем функции в программном модуле.

Когда переменная программного модуля используется в первый раз, создается новый экземпляр программного модуля. То есть новый набор внутренних переменных инициализируется таким образом, что разные переменные программного модуля используют различные наборы внутренних переменных.

Присваивание Программного Модуля

Программные модули рассматриваться как объекты – одна переменная программного модуля может присваиваться другой переменной программного модуля, что приводит к созданию новой ссылки на один и тот же экземпляр программного модуля. Другими словами, переменные программного модуля используют тот же набор внутренних переменных.

В данном примере будет создано два программных модуля:

DRAFT

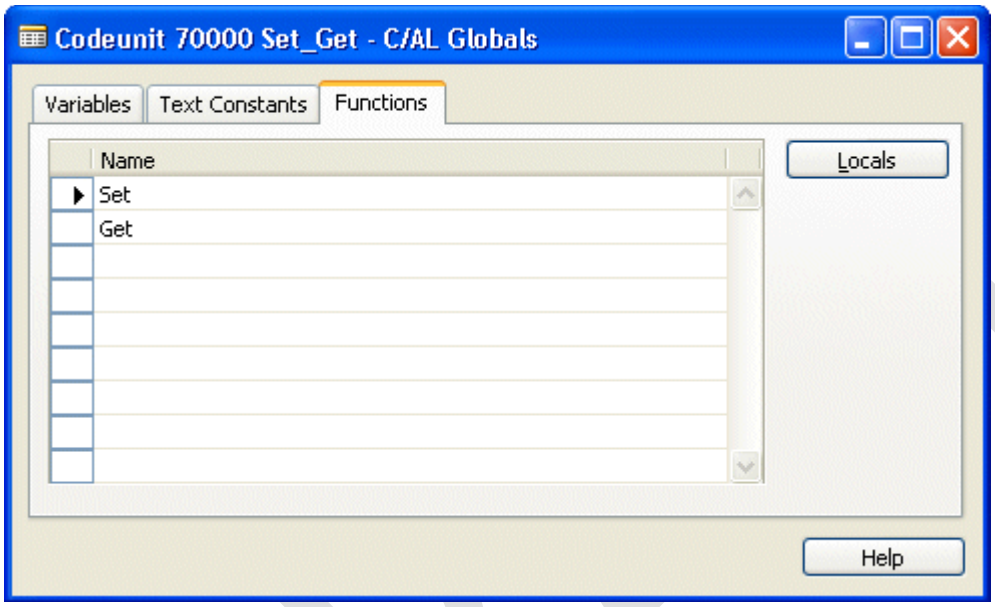
- Первый программный модуль имеет две функции Set и Get. Функция Set присваивает внутренней переменной значение переданного параметра. Функция Get возвращает значение присвоенное внутренней переменной.
- Второй программный модуль имеет две переменные, которые ссылаются на первый программный модуль. Затем одной переменной присваивается вторая переменная, таким образом, обе переменные ссылаются на один экземпляр программного модуля.

Чтобы создать первый программный модуль:

1. Создайте новый программный модуль и откройте для него окно ***C/AL Globals***.

2. Объявите переменную *InternalInt* с типом данных Integer.

3. В окне ***C/AL GLobals*** перейдите на закладку ***Functions*** и создайте две переменные: Set и Get.

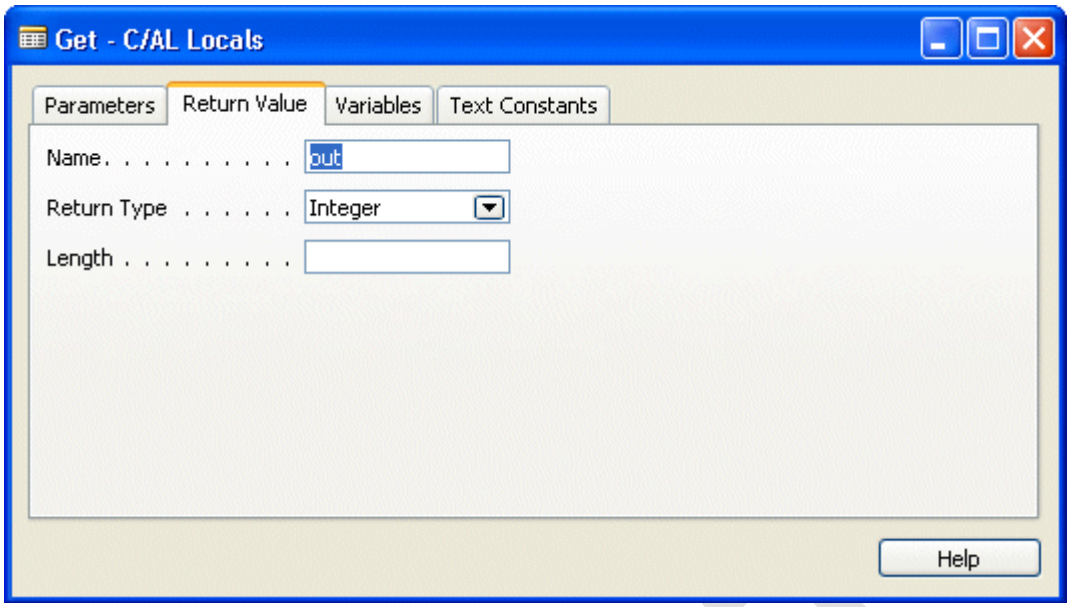


4. Выберите функцию Set и щелкните кнопку Locals, чтобы открыть окно ***C/AL Locals***.

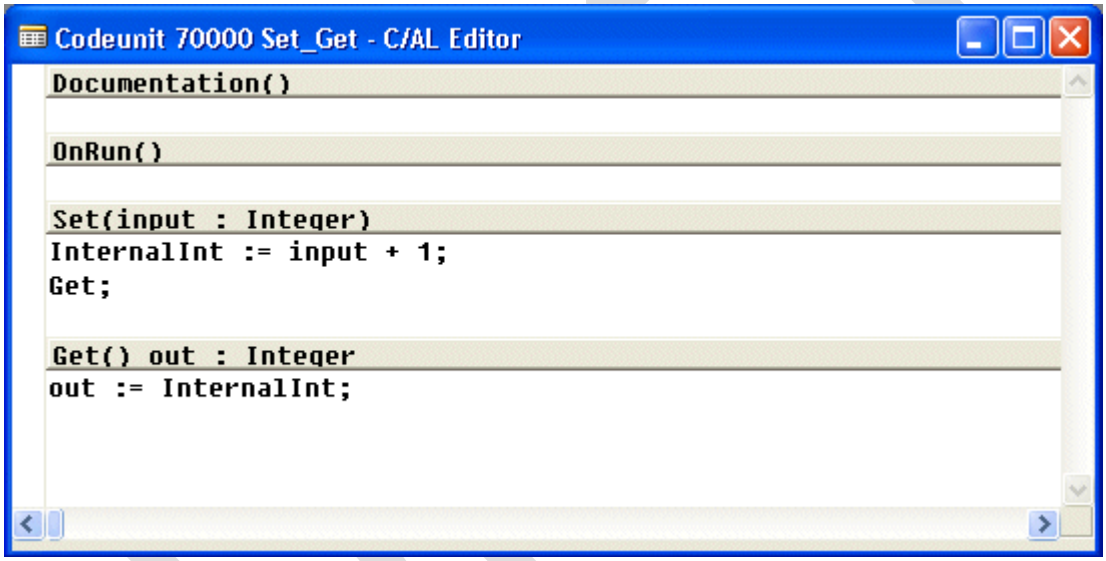
5. В окне ***C/AL Locals*** перейдите на закладку ***Parameters*** и создайте параметр *input* с типом данных Integer, затем щелкните кнопку OK.

6. В окне ***C/AL GLobals*** выберите функцию Get и щелкните кнопку Locals, чтобы открыть окно ***C/AL Locals***.

7. В окне ***C/AL Locals*** перейдите на закладку ***Return Value*** и создайте возвращаемое значение *out* с типом данных Integer, затем щелкните кнопку OK.



8. В окне C/AL Editor программного модуля появились два новых раздела – по одному для каждой добавленной функции. Введите в них следующий код на языке C/AL:



Функция Set увеличивает полученное значение на единицу, сохраняет его во внутренней переменной и вызывает функцию Get.

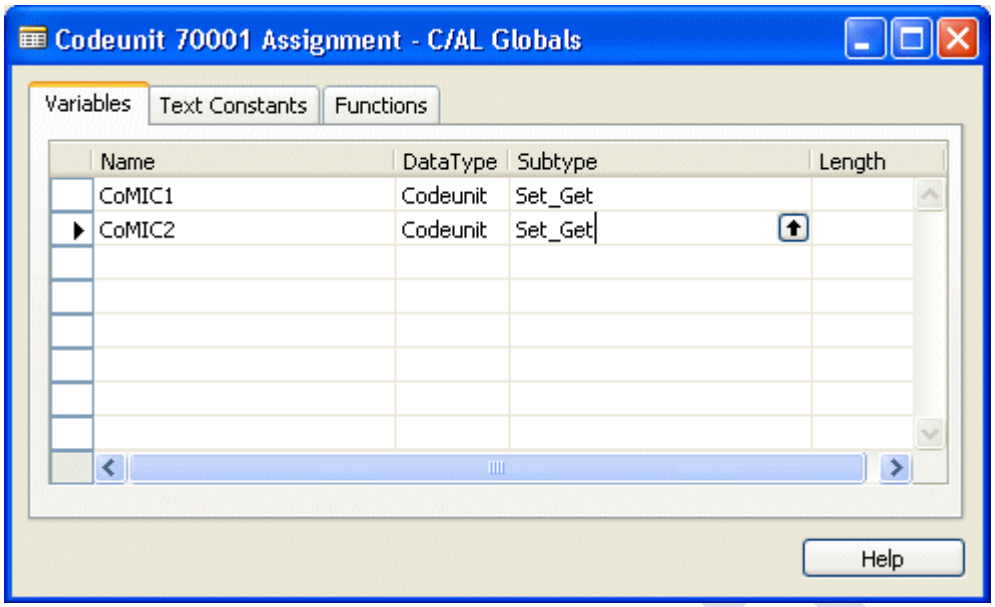
9. Сохраните и откомпилируйте программный модуль. В данном примере он был сохранен с идентификатором 70000 и названием ‘Set_Get’.

Следующим шагом будет создание программного модуля, который вызывает два экземпляра программного модуля Set_Get.

Чтобы создать второй программный модуль:

1. Создайте новый программный модуль и откройте окно ***C/AL Globals***.

2. В окне ***C/AL Locals*** создайте две следующие переменные:



Обе переменные связаны с программным модулем Set_Get.

3. Добавьте следующий код в триггер OnRun

```
CoMIC1.Set(1);

CoMIC2.Set(2);

//в два экземпляра программного модуля 70000 передаются соответственно значения 1 и 2

MESSAGE('When running two instances, the value of CoMIC1 is %1 and the value of CoMIC2 is %2.',CoMIC1.Get,CoMIC2.Get);

//Будут получены значения из программного модуля и отображены в сообщении.

//CoMIC2 вернет 2; ComIC2 вернет 3, так как к переданным параметрам была добавлена 1.


CoMIC2 := CoMIC1;

//переменной CoMIC2 присвоена переменная CoMIC1, теперь они используют один и тот же экземпляр программного модуля 70000.

MESSAGE('After assinging CoMIC2 to CoMIC1, the value of CoMIC1 is %1 and the value of CoMIC2 is also %2.',CoMIC1.Get,CoMIC2.Get);

//CoMIC1 и CoMIC2 покажут одинаковые значения.
```

В первой части приведенного кода вызываются два экземпляра программного модуля Set_Get и в него передаются значения, которые должны этими экземплярами использоваться. Далее одной переменной присваивается вторая переменная, теперь они используют один и тот же экземпляр программного модуля 70000 и по этой причине вернут одинаковые значения.

4. Сохраните и откомпилируйте программный модуль. В данном примере использовался идентификатор 70001 и название 'Assignment'.

Функция CLEAR в программных модулях

При использовании функции CLEAR по отношению к переменной программного модуля, имеющей ссылку на экземпляр программного модуля с двумя или более ссылками, эта функция удаляет только ссылку на программный модуль, а не на экземпляр программного модуля. Другими словами, программный модуль остается нетронутым и может использоваться другими переменными программного модуля, которым была присвоена ссылка на этот программный модуль. Чтобы

DRAFT

удалить экземпляр программного модуля, вы должны удалить все ссылки на программный модуль с помощью функции CLEAR. Если необходимо удалить внутренние переменные в программном модуле, вы должны вызвать функцию CLEARALL из определенной пользователем функции, находящейся внутри программного модуля. Когда локальная переменная программного модуля выходит за пределы области видимости, то есть больше не может использоваться программным модулем, то она автоматически удаляется.

Программный модуль Single Instance

В некоторых случаях существует только один экземпляр программного модуля. Это означает, что переменные определенного программного модуля будут использовать один и тот же набор переменных. Если значение свойства SingleInstance установлено в Да, все переменные данного программного модуля будут использовать один и тот же экземпляр, что позволяет разработчику создавать глобальные переменные.

> **Примечание**
>
> Рекомендуется избегать использования глобальных переменных в большинстве типов программного кода. Однако в определенных ситуациях может возникнуть необходимость их использования, например, чтобы убедиться в том, что используется только один экземпляр внешней переменной.

Единственному экземпляру программного модуля присваивается значение, когда вы используете его впервые. Обычные экземпляры программного модуля (программные модули, для которых свойство SingleInstance = Нет), удаляются в случаях, когда последняя переменная программного модуля, использующая данный экземпляр программного модуля, выходит за пределы области видимости. Тем не менее, программные модули Single Instance остаются действующими до тех пор, пока вы не закроете фирму (company).

> **Пример**
>
> Откройте созданный ранее программный модуль 70000 и установите в свойстве *SingleInstance* значение Да. Сохраните его с идентификатором 70002 и наименованием SingleInst.

Снова откройте программный модуль 70000 и создайте в нем две переменные:

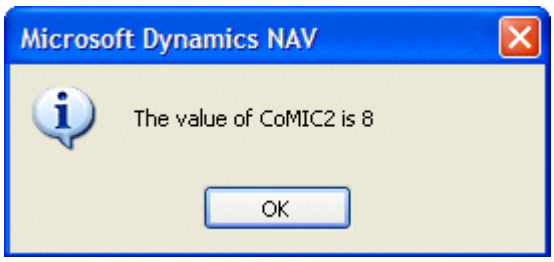
Variable	Data Type	Subtype
CoMIC1	Codeunit	SingleInst
CoMIC2	Codeunit	SingleInst

В программном модуле 70000 добавьте следующий код в триггер OnRun:

```
CoMIC1.Set(7);

//создан экземпляр программного модуля, если он не существовал ранее

CoMIC2.Get();

//возвращенное значение будет 8 (=7+1, добавленная функцией Set в программном модуле 70002)

//CoMIC2 использует тот же экземпляр, что и CoMIC1, поэтому они используют одни и те же внутренние переменные.

MESSAGE('the value of CoMIC2 is %1',CoMIC2.Get);
```

Теперь запустите программный модуль 70000, система покажет следующее сообщение:

DRAFT



К введенному программным модулем 70000 значению 7 в программном модуле 70002 была добавлена 1.

Теперь откройте программный модуль 70002 и измените код в функции Set, например, на

```
InternalInt := input+4
```

Закройте, сохраните и откомпилируйте программный модуль 70002.

Теперь запустите программный модуль 70000, и вы снова получите такое же сообщение как на рисунке выше. Так произошло потому, что программный модуль остается загруженным в памяти клиента, до тех пор, пока клиент не будет перезапущен.

> **Примечание**
>
> Программный модуль single Instance можно использовать в различных объектах, не обязательно в одном объекте.

Ограничения в Использовании Программных Модулей

К глобальным переменным и временным таблицам программного модуля нельзя получить непосредственный доступ из других прикладных объектов. Единственный способ получения доступа к этим значениям – использование функций, созданных вами в программном модуле.

В программном модуле могут использоваться все функции языка C/AL. Но обратите внимание на то, что вы не можете создать функцию с тем, же именем, что у встроенной функции. Кроме того, две и более функции, заданные пользователем, не могут иметь одинаковое имя (за исключением тех случаев, когда они являются частью разных прикладных объектов).

DRAFT

ГЛАВА 16 ВВЕДЕНИЕ В ЯЗЫК C/AL

В данной главе представлено введение в язык C/AL. В ней описано, как использовать язык C/AL для создания функций, а также приведено описание синтаксиса языка.

- Что можно делать с помощью языка C/AL?
- Что такое программный оператор, выражение и оператор?
- Введение в элементы выражения языка C/AL
- Язык управления C/AL

16.1 Что Можно Делать с Помощью Языка C/AL?

В предыдущих разделах данной книги вы узнали, как создавать базовые объекты, такие как таблицы и формы. Но недостаточно просто запускать эти объекты. Чтобы получить цельное приложение нужно заставить объекты базы данных работать вместе. Язык C/AL это тот клей, который объединяет объекты базы данных в одно целое.

При создании профессиональных приложений, часто требуются специальные функции. Язык C/AL позволяет создавать такие функции, расширяя, таким образом, функциональность среды C/SIDE. Например, можно создать специальные функции, которые будут использоваться в любом месте базы данных.

Но самыми важными возможностями предоставляемыми языком C/AL являются:

Создание Собственных Функций. Хотя среда C/SIDE содержит большое число встроенных функций, иногда необходимо (или удобнее) создать свои собственные функции. Например, может потребоваться создать собственные функции при разработке приложения, которое неоднократно использует нетривиальную обработку данных.

Объединение Объектов Базы Данных. C/AL код объединяет объекты базы данных вместе. C/AL включает набор функций, с помощью которых можно управлять взаимодействием объектов базы данных в приложении.

Чтение, Запись и Изменение Данных. Язык C/AL включает функции для чтения, записи и изменения данных в таблицах.

16.2 Что Такое Программный Оператор, Выражение и Оператор?

В данном разделе будет объяснены следующие термины:

- Оператор
- Выражение
- Тип Данных
- Операция

Представим следующий пример кода на языке C/AL:

Amount := 34 + Total;

Отдельная строка, содержащая код на языке C/AL также называется программный оператор (простая законченная инструкция, записанная в коде). В следующей таблице показаны ее составные элементы.

Элементы	Описание

DRAFT

Элементы	Описание
34 + Total	Выражение. В данном случае, выражение состоит из арифметического оператора (+) и двух операндов (34 и Total), которые в свою очередь также могут быть выражениями. Для каждого корректного выражение на языке C/AL может быть вычислено конкретное значение.
:=	Оператор присваивание. Когда для выражения в правой части будет вычислено, оно будет присвоено или помещено в переменную Amount.
Amount	Это переменная. Она ссылается на ячейку памяти, где хранится значение.

Что такое Выражение на Языке C/AL?

Выражение является основной концепцией языка C/AL. В данном разделе приведено описаний выражений и как они используются.

Выражения можно передавать в функции на языке C/AL в качестве параметров. Рассмотрим следующий программный оператор:

```
Date := DMY2DATE(31, 12, 2001)
```

Эта функция получает три выражения в качестве параметров: 31, 12 and 2001.

Выражение на языке C/AL это может состоять из данных, переменных, массивов, операций и функций, все это после вычисления становится значением с определенным типом данных.

Все выражения на языке C/AL строятся на:

- Константах
- Переменных
- Операциях
- Функциях

В зависимости от элементов, входящих в выражение, результат вычисления будет определенного типа данных. В следующей таблице приведены некоторые типовые выражения:

Выражение	Результат Вычисления
'Welcome to Hawaii'	Строка 'Welcome to Hawaii'
'Welcome' + ' to Hawaii'	Строка 'Welcome to Hawaii'
43.234	Число 43.234
ABS(-7234)	Число 7234
Len1 < 618	TRUE или FALSE в зависимости от значения переменной lenl

В первой в результате вычисления был получен такой же текст. Во второй строке была выполнена операция конкатенации двух строк. В третьей строке результатом стало такое же число, как указано в самом выражении. Четвертое выражение содержит функцию, которая преобразовала полученный параметр в 7234. В последней строке выполняется сравнение переменной и числовой константы.

Далее будет предоставлена детальная информация о типах данных в языке C/AL.

DRAFT

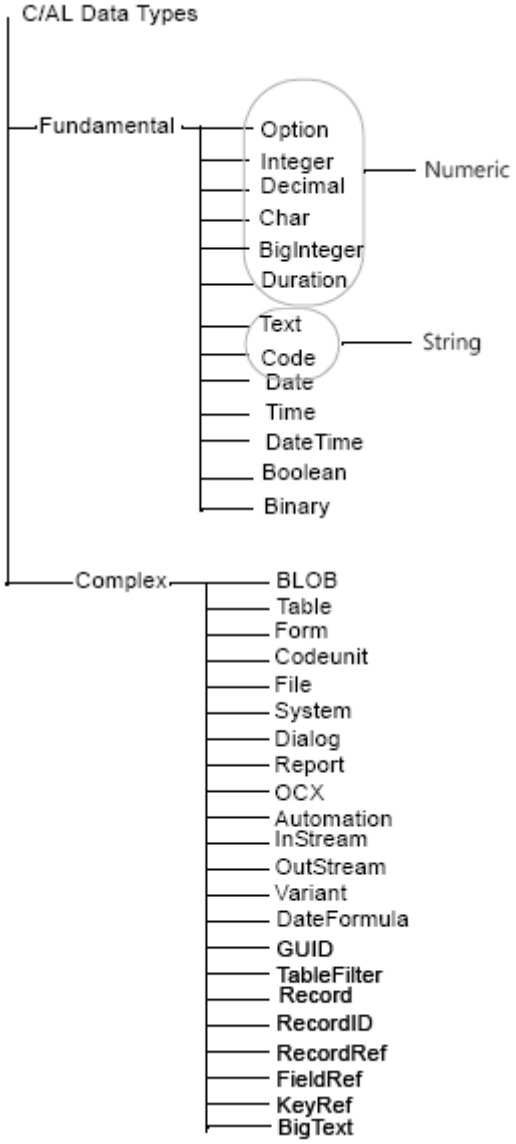
Введение в Типы Данных Языка C/AL

Как уже было показано, переменные могут использоваться для хранения значений различных типов данных. Определение переменных заранее позволяет:

- Создавать более быстрый код.
- Сохранять место.
- Избежать ошибок выполнения из-за переполнения.
- Избежать ошибок выполнения вызванного невозможностью преобразования типов.

Например, если известно, что переменная будет содержать значения от 0 до 700, значит, следует определить переменную типа integer вместо decimal. В результате все вычисления будут выполняться быстрее, так как система использует 4 байта для операций с целыми числами вместо 12 байт, которые требуются для decimal. С другой стороны нужно использовать такой тип данных, который может содержать все возможные результаты вычислений. Например, если попытаться сохранить значение 1233.345 в переменной с типом integer, то произойдет ошибка выполнения.

C/AL содержит набор типов данных. Эти типы данных можно разделить на следующие категории:



Базовые Типы Данных

C/AL включает базовые типы данных для хранения значений типа boolean, числа, текст, время и даты.

Boolean. Возможные значения TRUE и FALSE.

Integer. Используется для хранения целых чисел между -2,147,483,647 и 2,147,483,647.

BigInteger. Используется для хранения очень больших целых чисел.

Duration. Используется для представления разности между двумя значениями типа datetime в миллисекундах.

Option. Для представления опций. Значения типа Option могут быть легко конвертированы в числовые значения. Диапазон возможных значений составляет от -2,147,483,647 до 2,147,483,647.

Пример

Предположим, что Number это значение числового типа, а поле Type содержит набор опций в таблице ***Purchase Header***. Ниже показано преобразование значения типа option в число:

```
Number := "Purchase Header".Type
```

Пример

Следующий пример показывает, как использовать поле option в качестве константы в коде C/AL:

```
"Purchase Header".Type := "Purchase Header".Type::Invoice
```

Decimal. Используется для хранения числовых значений в диапазоне от $-10+E^{63}$ to $+10+E^{63}$

Date. Хранит даты в диапазоне с 01 Января 0000 (год 0) до 31 Декабря 9999. Неопределенная дата указана как 0D. Все даты имеют соответствующие закрывающие даты. Закрывающая дата трактуется системой как интервал после указанной даты и перед следующей нормальной датой.

Time. Хранит время. Неопределенное время указано как 0T. Корректно любое время в интервале от 00:00:00 до 23:59:59.999.

DateTime. Хранит дату и время.

Тип данных datetime хранится в базе данных как UTC. UTC это международный стандарт времени (ранее GMT). Значения datetime всегда отображаются в Dynamics NAV как локальное время. Локальное время определяется на основании часовой зоны и настроек перехода на зимнее время на компьютере пользователя.

В тип данных datetime всегда нужно вводить локальное время. При вводе локального времени в datetime оно конвертируется в UTC на основании текущих настроек компьютера.

Неопределенное datetime указывается как 0DT

C/SIDE Database Server

Самое ранее разрешенное значение datetime – 01 Января 0000, 00:00:00.000.

Самое позднее разрешенное значение datetime – 31 Декабря 9999, 23:59:59.999.

SQL Server

Самое ранее разрешенное значение datetime – 01 Января 1754, 00:00:00.000.

Самое позднее разрешенное значение datetime – – 31 Декабря 9999, 23:59:59.999.

Любые значения datetime за пределами данного диапазона, приведут к появлению сообщения об ошибке.

Неопределенные даты хранятся как 1 Января 1753, 00:00:00.000.

DRAFT

Char. Хранит один символ как значение от 0 до 255. Может быть конвертирован в число или символ. Это означает, что над типами данных char можно выполнять такие же математические операции как и над числовыми типами данных.

Пример

Переменной типа char присваивается строковая константа длинной в один символ:

C := "A"

Пример

Также можно присвоить переменной типа char один символ из переменной типа text, code или binary:

C := S[2]

Text. Хранит текстовые строки. Возможная длина текстовой строки составляет от 1 до 1024 символа. Можно обратиться к любому символу строки, используя конструкцию, например, A[65], которая вернет значение 65-го символа в переменной A. Результирующее значение будет типа данных char. Длина переменной типа text соответствует количеству символов в строке. У пустой строки длина 0.

Примечание

При использовании типов данных text или code важно различать максимальную длину строки и фактическую длину строки. Максимальная длина строки – это количество символов, которое можно поместить в строку, а фактическая длина – это сколько символов фактически используется в строке.

В следующей таблице приведены типовые примеры текстовых строк. Предполагается, что максимальная длина текстовой переменной в примере равна 6:

Присваивание	Результат
t := 'RnR!';	Переменная t теперь содержит значение "RnR!".
t := '123456abx';	В результате произойдет ошибка выполнения, так как длина (9) превышает максимальную длину (6).

Code. Хранит специальных вид текстовых строк. Когда текстовая строка присваивается переменной типа code, все символы строки переводятся в верхний регистр, пробелы, идущие перед и после текста, удаляются. Можно обратиться к любому символу строки, используя конструкцию, например, A[65]. Результирующее значение будет типа данных char. Максимальная длина переменной типа code составляет от 1 до 250 символов. Длина переменной типа code соответствует количеству символов в строке без учета пробелов в начале и конце строки.

В следующей таблице приведено несколько типовых примеров. Предполагается, что максимальная длина переменной типа Code составляет 4:

Присваивание	Переменная содержит значение	Длина
c := 'AbC';	'ABC'	3
c := '1';	'1'	1
c := '';	'' (пустая строка)	0 (ноль)
c := ' 2 ';	'2'	1
c := '1 2';	'1 2'	3

DRAFT

Комплексные Типы Данных

C/AL также содержит ряд комплексных типов данных. Комплексные типы данных используются для работы, например, с записями таблицы, растровыми изображениями или файлами на диске. Так как C/AL является объектно-ориентированным, то каждый комплексный тип данных может включать переменные и функции.

BLOB. Большой Двоичный Объект (Binary Large Object). Переменные этого типа данных отличаются от обычных числовых и строковых типов данных тем, что имеют переменную длину. BLOB используется для хранения текстов, растровых изображений или пользовательские данные. Максимальный размер BLOB составляет 2GB.

Record. Запись - комплексный тип данных, состоящий из нескольких простых элементов – полей (field). Запись (Record) соответствует строке таблицы. Каждое поле записи используется для хранения значений определенных типов данных. Доступ к полю (field) осуществляется через указание названия переменной ссылающейся на запись (record) точку и название поля (field). Запись (record) используется для хранения информации с фиксированным набором свойств.

Form. Форма. Переменная данного типа используется для хранения форм. Этот комплексный тип данных содержит ряд более простых элементов – элементы управления. Элементы управления используются для отображения информации или получения информации от пользователя.

Codeunit. Программный модуль. В данных переменных хранятся программные модули (codeunits). Этот комплексный тип данных может хранить набор пользовательских функций.

File. Переменная данного типа позволяет осуществлять доступ к файлам операционной системы.

Dialog. Переменные данного типа хранят диалоговые окна. Для них доступен ряд функций, позволяющих управляться диалоговыми окнами.

Report. Отчет. Переменные этого типа данных используются для хранения отчетов. Этот комплексный тип данных может содержать ряд более простых элементов – элементов управления. Элементы управления используются для отображения информации пользователю.

DateFormula. Этот тип данных используется для хранения формулы даты, имеет те же возможности что и параметр для функции CALCDATE. Тип данных DateFormula предоставляет многоязыковые возможности для функции CALCDATE.

GUID. В данном типе данных хранится глобально уникальный идентификатор (GUID) Используется для идентификации объектов, программ, записей и т.д. Главное свойство GUID заключается в том, что все значения уникальны. Значения генерируются по алгоритму, разработанному Microsoft, который гарантирует уникальность.

Стандартное текстовое представление GUID: {12345678-1234-1234-1234-1234567890AB}.

TableFilter. Этот тип данных используется для применения фильтров к другим таблицам.

В настоящее время, используется для хранения Фильтров Защиты в таблице Права Доступа (Permission).

RecordRef. Этот комплексный тип данный используется для идентификации строки в таблице. Каждая запись (record) состоит из полей (fields). Запись (record) используется для хранения информации с фиксированным набором свойств.

RecordRef может ссылаться на любую таблицу базы данных. Для выбора таблицы, к которой требуется получить доступ, используйте функцию RecordRef.OPEN. После использования функции RecordRef.OPEN будет создан новый объект. Этот объект содержит ссылку на открытую таблицу, фильтры, саму запись (record) и все ее поля (field).

Если одной переменной RecordRef присвоить другую переменную RecordRef они обе будут ссылаться на один и тот же экземпляр таблицы.

DRAFT

RecordID. Этот комплексный тип данных содержит номер таблицы и ее первичный ключ. RecordID можно хранить в базе данных, но по нему нельзя устанавливать фильтры.

FieldRef. Этот комплексный тип данных используется для идентификации поля в таблице и предоставляет доступ к данному полю. Переменная fieldref может ссылаться на любое поле любой таблицы базы данных.

KeyRef. Этот комплексный тип данных используется для идентификации ключа и полей, которые входят в данных ключ. Переменная keyref может ссылаться на любой ключ любой таблицы

InStream и OutStream. Переменные данного типа позволяют читать и писать в файлы или BLOB. Кроме того, InStream и OutStream можно использоваться для чтения и записи в объекты типов Automation и OCX.

Variant. Этот тип данных может содержать следующие типы данных: record, file, action, codeunit, Automation, boolean, option, integer, decimal, char, text, code, date, time, binary, DateFormula, TransactionType, InStream и OutStream.

BigText. Этот комплексных тип данных может содержать большие текстовые документы. Данные, содержащиеся в BigText нельзя отобразить в отладчике или в сообщении. Однако можно использовать предоставляемые BigText функции, чтобы извлечь часть текста и поместить его в обычную строку, которую можно отобразить.

Максимальная длина переменной BigText составляет 2147483647 символов, что эквивалентно 2 ГБ.

OCX и Automation. См. ГЛАВА 19 РАСШИРЕНИЕ ВОЗМОЖНОСТЕЙ ЯЗЫКА C/AL.

Дополнительную информацию о типах данных, см. в онлайновой справке *C/SIDE Reference Guide*.

Создание Массивов Переменных

Система позволяет создавать 10-размерные массивы на основании простых и комплексных типов данных. Система не ограничивает количество элементов, каждого измерения, но общее количество элементов не может быть больше 1 000 000. Физический размер ограничен 2 ГБ (или доступной оперативной памятью). Доступ к элементу массива осуществляется указанием его индекса для каждого измерения. Диапазон индексов бывает от 1 до размера измерения. Если указать значение индекса, лежащего за пределами измерения, произойдет ошибка выполнения.

Пример

Предположим, что Foo это массив типа Integer с одним измерением и размерностью 10.

Чтобы получить доступ к первому элементу следует использовать Foo[1]. Для доступа к последнему элементу используйте Foo[10].

Пример

Предположим, что Bar это массив типа Date размерностью 2x3x4. Массив Bar включает 24.

Чтобы получить доступ к первому элементу следует использовать Bar[1,1,1]. Для доступа к последнему элементу используйте Bar[2,3,4].

16.3 Введение в Выражения на Языке C/AL

В предыдущем разделе было приведено краткое описание выражений на языке C/AL и типов данных. Цель данной главы дать представление об основных элементах выражений на языке C/AL. Далее будет рассмотрены следующие темы:

- Константы
- Переменные
- Операции
- Функции

DRAFT

Константы

Константы это простейшие типы операндов, используемых в C/AL. Значение констант не изменяется в процессе выполнения кода. Константы могут быть определены для каждого типа данных языка C/AL.

> **Ввод Значений C/SIDE**
>
> Числовые значения, такие как 2,147,483,647 и 999,999,999,999,999.99 нельзя вводить в систему C/AL в таком виде. Запятые используются для удобочитаемости данного документа. Если ввести значения в C/AL editor с запятыми, то при компиляции произойдет ошибка.

boolean константы. Константы типа Boolean могут быть представлены значениями: TRUE или FALSE.

integer константы. Константы типа Integer могут быть в диапазоне от -2,147,483,647 до 2,147,483,647.

decimal константы. Константы типа Decimal должны включать точку, для отделения дробной части «.» (зависит от региональных настроек) и иметь как минимум одно число после точки (например, «0»). Константы типа decimal могут использоваться для представления десятичных чисел между -999,999,999,999,999.99 и 999,999,999,999,999.99.

date константы. Константа типа date состоит из шести или восьми чисел, за которыми следует буква "D" (константа, содержащая неопределенную дату, вводится как "0D"). Формат даты может быть следующим DDMMYY или DDMMYYYY.

time константы. Константа типа time состоит из шести или девяти чисел, за которыми следует буква 'T' (константа, содержащая неопределенную дату, вводится как "0T"). Константы вводятся в следующем формате HHMMSS[.XXX], время в формате 24 часа, где необязательная часть содержит тысячные доли секунды.

text константы. Константа типа text содержит строку символов. Среда C/SIDE позволяет указать для текстовых констант уникальные идентификаторы, далее с помощью этих идентификаторов можно обращаться к определенным текстовым константами. Примером текстовых констант являются сообщения об ошибках и предупреждения.

В следующей таблице приведены примеры различных типов констант, использующихся в языке C/AL:

Константа	Описание
TRUE	Константа типа Boolean
50000	Константа типа Integer
-23.7	Константа типа Decimal
211201D	Константа типа Date (21 Декабря 2001)
141230T	Константа типа Time (время 14:12:30)
‘ABC’	Константа типа Text

Использование Переменных в Языке C/AL

В языке C/AL существует для вида переменных: переменные, определенные пользователем и системные переменные.

Переменные, определенные пользователем

Переменные, определенные пользователем – это те переменные, которые вы определяется при создании кода на языке C/AL. Переменные можно определить как глобальные и использоваться для всех функций программного модуля или как локальные и использовать только в одной функции.

DRAFT

Эти переменные можно использовать для хранения информации в процессе работы и изменять их значения как угодно.

Системные Переменные

Системные переменные, предоставляются и поддерживаются системой. К системным переменным относятся: Rec, xRec, CurrForm и CurrReport.

При работе системы, она выполняет код в функциях и триггерах, например, выполняется обработка записей таблицы. Перед тем как выполнять код, система автоматически присваивает значения системным переменным, и эти значения могут в дальнейшем использоваться в триггерах и локальных функциях.

При выполнении триггеров и функций, системные переменные используются так же как обычные переменные (им могут быть присвоены новые значения). Это происходит потому, что система не обновляет данные переменные при выполнении кода C/AL, а обновляет только перед тем как приступить к выполнению функций и триггеров.

Наименования Переменных

Существует ряд правил и ограничений при задании имени переменной:

- Имена переменных должны быть уникальными – программный модуль не может содержать две переменные с одинаковыми именами.
- Переменные, определенные пользователем, не могут иметь такие же имена, как и системные переменные.
- Система не различает строчные и прописные буквы, поэтому Smith и SMITH будет ссылать на одну переменную.
- В C/AL можно использовать специальные символы в названиях переменных (например, пробелы).
- Максимальная длина наименования переменной составляет 30 символов.
- Имя переменной не может совпадать с названием функции C/AL или зарезервированным словом. Обратите внимание, что данное правило применяется как к написанию прописными буквами, так и к написанию строчными буквами. Пример, нельзя использовать ни вариант BEGIN, ни begin.

Разрешены все символы ASCII, кроме:

- Управляющих символов (ASCII 0-31, 255)
- Символа " (ASCII 34)

Чтобы использовать все комбинации символов, нужно заключить имя переменной в двойные кавычки, например: "Customer No.". Если вы не хотите использовать двойные кавычки, то имя переменной должно:

Первый символ должен быть:

- Буквой в диапазоне: a..z, A..Z (ASCII 97-122, 65-90), или
- Символом подчеркивания (ASCII 95),

...остальные 29 символов могут быть

- Буквой в диапазоне: a..z, A..Z (ASCII 97-122, 65-90),
- Символом подчеркивания (ASCII 95), или
- Цифрой в диапазоне 0..9 (ASCII 48-57).

Как было указано ранее, в наименование переменной можно включить один или несколько специальных символов (пробелов и т.п.). Однако в этом случае, имя переменной должно быть

DRAFT

полностью заключено в двойные кавычки. В этом случае имя переменной может содержать смесь букв, цифр и специальных символов.

> **Примечание**
>
> Двойные кавычки не являются частью имени переменной, но они необходимы, чтобы избежать появления ошибки при компиляции программного модуля.

Ниже приведены примеры корректных имен переменных:

- Customer
- MegaMan1
- "@Vendor"
- "1st AddressLine"
- "Purchase/Sales"
- "Sales In GBP"
- " YesCrazy Name1Ñ3"

...а далее приведены некорректные имена переменных

- 34467
- 23"Tubes
- Stock Group4
- "Sale"s in GBP"
-)-Names
- END

Инициализация

Перед выполнением C/AL кода переменные автоматически инициализируются. В переменные типа boolean устанавливается значение FALSE, в числовые переменные – 0, в текстовые – '' (пустая строка), в переменные date и time устанавливаются соответственно значения 0D и 0T.

Как упоминалось выше, система автоматически обрабатывает системные переменные. Сюда также относится необходимая инициализация. Это значит, что от пользователя не требуется выполнение никаких действий, перед тем как он сможет использовать системные переменные.

Присваивание и Преобразование Типов

Существует два способа выполнить присваивание значений:

- Присваивание параметров, переданных в функцию, например: FUNCTION(Expression). Результат вычисления выражения должен быть соответствующего типа данных, или быть такого типа данных, который может быть автоматически преобразован в соответствующий тип данных. См. ГЛАВА 25 ПРЕОБРАЗОВАНИЕ ТИПОВ ДАННЫХ.
- С помощью оператора присваивания ":=" (Например, Variable := Expression). Результат вычисления выражения в правой части, должен такого же типа данных, как и переменная (слева от оператора присваивания) или же быть такого типа данных, который может быть автоматически преобразован в тип данных переменной.

Автоматическое преобразование типов выполняется:

- Если параметр, переданный в функцию, не принадлежит к корректному типу данных. Это может произойти, если в функцию, которая должна вызываться с параметром типа integer был передан параметр типа decimal.

DRAFT

- Результат вычисления выражения справа от оператора присваивания принадлежит к другому типу данных, чем переменная (слева от оператора присваивания).

Автоматическое преобразование типов легко выполняется между числовыми типами данных (если совпадает размерность):

char ↔ integer ↔ decimal

Автоматическое преобразование типов данных также легко выполняется между строковыми типами данных:

code ↔ text

Предыдущие примеры относились к простым переменным. Тем не менее, такие же правила применяются для массивов.

Тип данных Text можно легко преобразовать в bigtext, но тип данных bigtext должен быть разделен на несколько меньших частей, перед тем как он сможет быть конвертирован в text.

text ↔ bigtext

Примечание

При выполнении преобразования в процессе присваивания все равно может произойти ошибка выполнения, даже если типы являются совместимыми. Ошибка выполнения произойдет, если преобразованное значение находится за пределами разрешенного диапазона для переменной слева от операции присваивания. То же самое относится и к функциям, для которых выполняется преобразование переданных параметров.

Пример

Переменная B определена как массив с одни измерением, включающим четыре элемента типа text с максимальной длиной 15. Второму элементу массива можно присвоить значение следующим образом:

```
B[2]:= 'Enter your name'
```

Пример

Result это переменная типа option, а Amount и Total переменные типа decimal.

Рассмотрим следующий код:

```
Amount := 10;
Total := 4;
...
Result := Amount + Total;
```

Этот код всегда можно будет откомпилировать, но при работе возможно возникновение ошибки выполнения, если результат выражения "Amount + Total" окажется за пределами разрешенного диапазона для переменной Result, т.е. за пределами диапазона с -2,147,483,647 по 2,147,483,647.

Корректное присваивание

В следующих таблицах показано, когда возможно выполнить присваивание переменной, заданного типа данных, результата вычисления выражения такого же или другого типа данных. Данные таблицы относятся только числовым и строковым типам данных.

Числовые Типы Данных:

Тип переменной	Тип выражения					
	Char	Option	Integer	Biginteger	Duration	Decimal

Тип переменной	Тип выражения					
	Char	**Option**	**Integer**	**Biginteger**	**Duration**	**Decimal**
Char	●	○	○	○	○	○
Option	●	●	●	○	○	○
Integer	●	●	●	○	○	○
Biginteger	●	●	●	●	●	○
Duration	●	●	●	●	●	○
Decimal	●	●	●	○	○	●

Строковые Типы Данных

Тип переменной	Тип выражения		
	Text	**Code**	**Bigtext**
Text	○	○	○
Code	○	○	○
Bigtext	○	○	○

● - присваивание корректно

○ - присваивание корректно, но возможно переполнение

Примечание

Переменная Bigtext может быть присвоена с помощью соответствующих функций, например, GETSUBTEXT.

Операторы Языка C/AL

В данном разделе приведено описание функций операторов языка C/AL:

Оператор C/AL	Предназначен
.	Доступ к полям записи, элементам управления формы или отчета
( )	Круглые скобки
[ ]	Ввод индексов
::	Область
+	Сложение
-	Вычитание
*	Умножение
/	Деление
DIV	Целочисленное деление
MOD	Модуль
>	Больше чем

DRAFT

Оператор C/AL	Предназначен
>=	Больше чем или равен
<	Меньше чем
<=	Меньше чем или равен
=	Равен
<>	Не равен
IN	В диапазоне
AND	Логическое И
OR	Логическое ИЛИ
NOT	Логическое НЕ
XOR	Исключающее логическое ИЛИ
..	Диапазон

Операторы "+" и "-" могут использоваться и как унарные и как бинарные операторы. Оператор "NOT" является унарным оператором. Все остальные операторы бинарные.

Большинство операторов можно использовать для различных типов данных. Действие этих операторов зависит от типов данных в выражениях, в которых эти операторы используются. Далее приведено несколько типовых примеров:

Пример

Оператор "+" как бинарный оператор:

number + number

Результатом будет сумма двух чисел, которая также будет являться числом.

Пример

Оператор "+" как бинарный оператор:

string + string

Результатом будет конкатенация строк, которая также будет являться строкой.

Пример

Оператор "+" как унарный оператор для отображения знака:

+ 34545

Приоритет Операторов

Для операторов существует определенный приоритет. Он задает порядок, в котором выполняются операции в данном выражении. Ниже приведена последовательность выполнения операторов C/AL:

1.(поля записи), [] (индексирования), () (скобки), :: (область)

2 NOT, - (унарный), + (унарный)

3 *, /, DIV, MOD, AND, XOR

4 +, -, OR

5 >, <, >=, <=, =, <>, IN

6.. (диапазон)

DRAFT

В следующих примерах показано как работает приоритет операций на практике.

Хотя выражения выглядят похоже, результат их выполнению будет различаться.

Пример

Результатом вычисления выражения 2 + 3 * 4 будет значение 14,

А результатом вычисления выражения (2 + 3) * 4 будет значение 20.

Вызовы Функций

Язык C/AL включает ряд функций, которые можно использовать для различных целей, таких как обработка строки, форматирование текста, работа с базой данных и т.п. Некоторые из этих функций могут использовать различные параметры.

При вызове функции параметры разделяются запятыми, необязательные параметры можно опускать, начиная справа. Это значит, что если для функции предусмотрено три необязательных параметра, то нельзя опустить второй параметр, не опустив третий.

Пример

Функцию

FUNCTION([Optional1] [, Optional2] [, Optional3])

Можно вызвать как:

FUNCTION(Optional1, Optional2)

Но нельзя вызвать:

FUNCTION(, Optional2, Optional3)

Пример

ABS это типичная функция C/AL с фиксированным числом параметров (1).

Value := -1033 {отрицательное целое число}

PositiveValue := ABS(Value) {результатом будет 1033}

Пример

Функция DMY2DATE это типичная функция, которую можно вызывать с различным числом параметров.

NewDate := DMY2DATE(5, 11, 1992)

{Возвращает дату 05 Ноября 1992}

В функцию DMY2DATE можно передать, 1, 2 или 3 параметра. Второй и третий параметры не обязательны. Если второй и третий параметр не используются, система использует системную дату в качестве значения по умолчанию.

16.4 Язык Управления C/AL

В данном разделе описаны базовые структуры языка управления C/AL, а также приведено описание их использования. Все программы на языке C/AL состоят из одного или нескольких программных операторов и выполняются последовательно сверху вниз. Однако часто нужно выполнять последовательностью выполнения операторов. Некоторые выражения должны выполняться несколько раз, а выполнение других выражений зависит от выполнения некоторых условий.

Структуры языка управления C/AL разделены на следующие основные группы:

- Составные Операторы
- Условные Операторы

DRAFT

- Операторы циклов
- Оператор WITH

Составные Операторы

С помощью синтаксиса языка C/AL можно использовать только по одному программному оператору. Однако если требуется выполнить несколько программных операторов, то они должны быть преобразованы в составные операторы. Для этого их нужно заключить между зарезервированных словами BEGIN и END. Синтаксис:

BEGIN

<Statement 1>;

<Statement 2>;

..

<Statement n>;

END

Программные операторы разделяются точкой с запятой. Лишняя точка с запятой перед END не приведет к ошибке, так как компилятор проинтерпретирует ее как пустой программный оператор.

Структура BEGIN END также называется блоком. Блоки крайне полезны, при использовании с другими структурами, которые будут рассмотрены далее.

Условные Операторы

При использовании условных операторов осуществляется проверку условия и выполняется (в зависимости от его значения) один или несколько операторов. В языке C/AL два типа условных операторов:

1 IF THEN [ELSE], выбор из двух вариантов.

2 CASE, можно выполнить выбор более чем из двух вариантов.

Условный Оператор IF THEN ELSE

Для условного оператора IF THEN ELSE используется следующий синтаксис:

IF <Condition> THEN <Statement1> [ELSE <Statement2>]

который означает:

Если <Condition> истинно, то выполняется <Statement1>. Если <Condition> ложно, то выполняется <Statement2>.

Как было указано раньше, квадратные скобки вокруг ELSE <Statement2> указывают на то, что эта часть оператора не обязательна.

Данный оператор используется, если в зависимости от результата вычисления <Condition> нужно выполнять различные действия.

Используя вложенные структуры IF THEN ELSE можно создать более сложные операторы. Типичный пример:

IF <Condition1> THEN IF <Condition2> THEN <Statement1> ELSE <Statement2>

Если <Condition1>=FALSE, не никакой код выполняться не будет. Если <Condition1>=TRUE и <Condition2>=TRUE, то будет выполнено <Statement1>. Если <Condition1>=TRUE, а <Condition2>=FALSE, выполняется <Statement2>.

Обратите внимание, что перед ELSE нельзя ставить точку с запятой.

DRAFT

Чтение вложенных операторов IF THEN ELSE может оказаться сложным, при этом следует помнить правило, что ELSE связано с последним IF, для которого не указан ELSE.

Ниже приведено несколько примеров операторов IF THEN ELSE:

Пример

Оператор IF без необязательной части ELSE:

IF Amount < 1000 THEN Total := Total + Amount;

Example

(1)...

(2) IF Amount < 1000

(3) THEN BEGIN

(4) IF I > J THEN Max := I

(5) ELSE Max := J;

(6) Amount := Amount * Max;

(6) END

(7) ELSE

(8)...

Распространенной ошибкой, которую делают начинающие C/AL-программисты, заключается в установке лишней точки с запятой перед ELSE (строка 4). Это неправильный синтаксис языка C/AL, так как точка с запятой используется рак разделитель операторов, а строка 4 находится внутри оператора IF.

Условный Оператор CASE

Синтаксис оператора CASE:

CASE <Expression> OF

<Value set 1> : <Statement 1>;

<Value set 2> : <Statement 2>;

...

...

<Value set n> : <Statement n>;

[ELSE <Statement n+1>]

END;

В качестве <Expression> нельзя использовать запись (record), а <Value set> должно быть выражением или диапазоном.

Набор значений (value set) подобен набору констант (constant set), участвующему в операции IN, за исключением того, что не стоит в квадратных скобках. Таким образом, он может содержать одно значение, несколько значений, разделенных запятыми, диапазоны, несколько диапазонов, разделенных запятыми и т.п. Все значения, фигурирующие в наборе значений, должны иметь тип, совместимый с типом CASE-выражения.

Заметим, что блок ELSE для оператора CASE носит необязательный характер, а потому в синтаксисе заключен в квадратные скобки.

DRAFT

Когда оператор CASE выполняется, сначала вычисляется CASE-выражение. Это выражение иногда называют селектором (selector). Затем по очереди вычисляется каждое значение в каждом наборе значений.

Если одно из значений в первом наборе соответствует значению CASE-выражения, то выполняется первый оператор. Если одно из значений во втором наборе соответствует значению CASE-выражения, то выполняется второй оператор. Так продолжается до тех пор, пока не выполнится один из операторов, представленных в CASE.

Если проанализирован последний набор значений и ни один из операторов не был выполнен, то ищется ELSE-блок. Если он не пуст, то выполняется находящийся в нем оператор. Если же ELSE-блок отсутствует, то для данного оператора CASE не выполняется ни одного оператора.

Обратите внимание, что выполняться будет только один из операторов в вариантах CASE. Если одно и то же значение соответствует разным наборам значений в CASE, то выполняется оператор только для первого из них (кто идет раньше).

Пример

Следующий C/AL код отображает различные сообщения в зависимости от значения Number. Если значение Number не соответствует ни одному элементу структуры CASE, то используется элемент ELSE.

```
CASE Number OF

1,2,9: MESSAGE('1, 2 или 9.');

10..100: MESSAGE('Входит в диапазон от 10 до 100.');

ELSE MESSAGE('Не 1, не 2, не 9, и не входит в диапазон от 10 до 100.');

END
```

Использование Операторов Цикла

C/AL предлагает следующие механизмы для организации циклов:

- FOR, повторяет содержащийся в нем оператор пока переменная-счетчик не достигнет определенного для нее минимума или максимума.
- WHILE, повторяет содержащееся в нем выражение, пока указанное условие истинно (TRUE). Цикл может быть выполнен 0 и более раз.
- REPEAT, повторяет содержащееся в нем выражение, пока указанное условие не станет истинно. Цикл выполняется как минимум один раз.

Структура FOR TO/DOWNTO

Синтаксис для оператора FOR TO (и FOR DOWNTO):

FOR <Control Variable> := <Start Number> TO <End Number> DO <Statement>

Типы данных для <Control Variable>, <Start Number> и <End Number> должны быть boolean, числовой тип, time или date.

Оператор FOR используется, если нужно выполнить некий код определенное количество раз.

Переменная <control variable> указывает сколько раз должен быть выполнен код. Переменная <Control Variable> может увеличиваться или уменьшаться на единицу, в зависимости от того используется TO или DOWNTO.

Когда определяется тип переменно <Control Variable>...

При выполнении оператора FOR значения <Start Number > и <End Number> преобразуются при необходимости в такой же тип данных как <Control Variable>. Данное преобразование может вызвать ошибку выполнения.

DRAFT

Оператор FOR TO ни разу не выполнит оператор <Statement>, если <Start Number> больше чем <End Number>. Соответственно, оператор <Statement> не будет выполнен оператором FOR DOWNTO, если <Start Number> меньше чем <End Number>.

Примечание

Обратите внимание, что переменная цикла не должна изменяться внутри самого цикла FOR, иначе это может привести к непредсказуемым результатам. Кроме того, значение переменной цикла за его пределами (после завершения цикла FOR) считается неопределенным.

Пример

Создайте следующие переменные:

Variable	Data Type
Count	Integer

Оператор FOR использует переменную Count с типом данных integer.

```
FOR Count := 1000 TO 100000000000000 DO
```

При выполнении данного выражения возникнет ошибка, так как система попытается преобразовать начальное и конечное значения к тому же типу данных, что и переменная Count. Так как переменная Count была объявлена как Integer, то при попытке преобразовать значение 100000000000000 произойдет ошибка выполнения потому, что данное значение не входит в допустимый диапазон значений для типа данных Integer.

Пример

Этот пример демонстрирует вложенные операторы FOR.

Создайте следующие переменные;

Variable	Data Type
I	Integer
J	Integer
A	Integer

Установите в свойство *Dimensions* переменной A значение 5;7.

Следующие операторы FOR используются для инициализации всех элементов массива 5x7 значением 23.

```
FOR I := 1 TO 5 DO
  FOR J := 1 TO 7 DO
    A[I,J] := 23;
```

Структура WHILE DO

Для оператора WHILE DO используется следующий синтаксис:

WHILE <Condition> DO <Statement>

Если <Condition> = TRUE, оператор <Statement> будет выполняться до тех пор, пока <Condition> не станет равен FALSE. Если <Condition>= FALSE изначально, оператор <Statement> никогда не будет выполнен.

Оператор WHILE DO используется, когда нужно повторять выполнение какого-либо кода пока условие истинно (TRUE).

DRAFT

Пример

Создайте следующую переменную:

Variable	Data Type
i	Integer

Следующий C/AL код увеличивает значение переменной *i* до тех пор, пока оно не станет равно 1000, в конце отображается сообщение:

```
WHILE i < 1000 DO i := i + 1
Message(format(i))
```

Структура REPEAT UNTIL

Синтаксис оператора REPEAT UNTIL:

REPEAT <Statements> UNTIL <Condition>

<Statements> будет выполняться до тех пор, пока <Condition> не станет TRUE.

С первого взгляда это похоже на структуру WHILE. Но так как условие для оператора REPEAT UNTIL проверяется в конце, то заключенный в операторе код будет выполнен как минимум один раз. Этим он отличается от структуры WHILE, в которой проверка выполняется перед выполнением заключенного в операторе кода, это значит, что если первая проверка условия выявила <Condition> =FALSE, то код не будет выполнен ни разу.

Пример

Создать следующие переменные:

Variable	Data Type	Subtype
Count	Integer	
Customer	Record	Customer

Далее приведен типичный пример использования REPEAT UNTIL:

```
Count := 0;
IF Customer.FIND('-') THEN
  REPEAT
    Count := Count + 1;
  UNTIL Customer.NEXT <= 0;
Message('Таблица Клиенты содержит %1 записей.',Count);
```

Этот код использует цикл REPEAT UNTIL, чтобы подсчитать количество записей в таблице ***Customer***. Функция FIND находит первую запись. Каждый раз при вызове функции NEXT выполняется переход на следующую запись. Когда NEXT = 0 значит, что в таблице больше нет записей. Система покидает цикл и отображает сообщение, в котором указано количество найденных операций.

Оператор EXIT

Оператор EXIT используется для управления последовательностью выполнения. Синтаксис оператора EXIT:

EXIT([<Value>])

DRAFT

Оператор EXIT используется для остановки исполнения триггера. Он может применяться подобным образом и для циклов и аналогичных структур. Если вызов функции происходит внутри выражения, то требуется, чтобы она возвращала значение. Когда вы пишете собственную функцию, которая возвращает значение, то можете выполнять это с помощью оператора EXIT(Value).

Использование в локальной функции оператора EXIT без параметра соответствует использованию параметра 0. В этом случае функция на языке C/AL вернет значение 0 или '' (пустая строка).

Если оператор EXIT вызывается с указанием возвращаемого значения, то в следующих случаях произойдет ошибка компиляции:

- В системных триггерах.
- В локальных функциях, которые не поддерживают возвращаемые значения.

Пример

В следующем примере, показано, как использовать оператор EXIT в локальной функции. Предположим, что оператор IF используется для обнаружения ошибок. Если встречается ошибка, то выполнение прекращается, а локальная функция возвращает код ошибки 1.

```
FOR I := 1 TO 1000 DO BEGIN
  IF Amount[I] < Total[I] THEN EXIT(1);
  A[I] := Amount[I] + Total[I];
END;
```

Оператор WITH

Синтаксис оператора WITH:

WITH <Record> DO <Statement>

При работе с записями, обращение к полям выполняется следующим образом: <Record>.<Field>

Если предполагается большой объем работы с одной и той же записью, можно использовать оператор WITH. При использовании оператора WITH нужно указать наименование переменной типа запись (record) только один раз.

Внутри оператора <Statement>, к полям (fields) записи (<Record>) можно обращаться без указания наименования записи (record).

Можно использовать несколько вложенных операторов WITH. Если используются одинаковые имена, то внутренний WITH переопределяет внешний оператор WITH.

Пример

В данном примере показано два способа написания одного кода по созданию записи, которая позже может быть помещена в базу данных.

Создайте следующие переменные:

Variable	Data Type	Subtype
CustomerRec	Record	Customer

```
CustomerRec."No." := '1234';
CustomerRec.Name := 'Windy City Solutions';
CustomerRec."Phone No." := '555-444-333';
CustomerRec.Address := '1241 Druid Avenue';
```

DRAFT

```
CustomerRec.City := 'Windy City';
MESSAGE('Для данного клиента была создана переменная.');
```

Другой способ записи этого же кода:

```
WITH CustomerRec DO BEGIN
  "No." := '1234';
  Name := 'Windy City Solutions';
  "Phone No." := '555-444-333';
  Address := '1241 Druid Avenue';
  City := 'Windy City';
  MESSAGE('Для данного клиента была создана переменная.');
END;
```

Как Добавлять Комментарии к Вашим Программам

К коду можно добавить комментарии, а также можно «закомментировать» часть кода, чтобы он не выполнялся.

Существует два способа создания комментариев:

- Использовать "//" чтобы добавить одну строку комментария. Когда компилятор встречает символы "//" он игнорирует оставшуюся часть строки.
- Использовать "{" и "}" чтобы указать начало и конец блочного комментария.

Пример

```
{
Это пример комментария. Он будет проигнорирован компилятором C/AL
}
// Это тоже пример комментария
{ Этот комментарий { частично находится внутри } другого комментария }
```

Пример

Этот пример показывает, как не нужно делать:

```
A := 34;
B := 56;  {******************
C := 345;  * Не делай так!  *
D := 781; ******************}
```

Система решит, что третья и четвертая строки являются частью комментария. Поэтому присваивание будет выполнено для переменных A и B, и не будет для C и D. Вместо этого стоит использовать строчные комментарии:

```
A := 34;
B := 56;  //******************
```

DRAFT

```
C := 345; //*   Делайте так!  *
D := 781; //*******************
```

DRAFT

ГЛАВА 17 ИСПОЛЬЗОВАНИЯ ЯЗЫКА C/AL

В данной главе будут рассмотрены некоторые аспекты использования языка C/AL. В первом разделе будет представлены советы по использованию системных переменных. Во втором будет описано, как обрабатывать функции, которые могут вызвать ошибку выполнения, в зависимости от способа их использования. И последний, самый большой раздел будет содержать обзор функция языка C/AL и примеры их использования. В данный раздел включены наиболее часто используемые функции, поняв которые вы сможете создавать сложные приложения C/SIDE.

- Обзор
- Системные переменные
- Обработка ошибок выполнения
- Основные функции языка C/AL

17.1 Обзор

В данной главе описано, как использовать C/AL. Здесь вы узнаете, куда нужно помещать ваш код.

Где Писать C/AL Код

Как было сказано ранее, почти каждый объект среды C/SIDE содержит триггеры, в которые можно помещать свой C/AL код. Существуют триггеры для:

- Таблиц (Tables)
- Полей таблиц (Table fields)
- Форм, в т.ч. формы запросов для отчетов (Forms, request options forms)
- Элементы управления форм (Form controls)
- Отчеты (Reports)
- Элементы данных (Data items)
- Разделы (Sections)

Инициировать выполнение C/AL кода можно из:

- Командных кнопок (Command buttons)
- Элементов меню (Menu items)

Также можно поместить C/AL код в программный модуль и вызывать его из других мест, упоминавшихся выше.

Как видите, поместить код можно в различные места, и вызывать его для выполнения различными способами. Однако место, куда вы будете помешать свой код не должно выбираться случайно.

Ниже приведено несколько простых рекомендаций, которым необходимо следовать:

- В целом, код нужно помещать как можно ближе к объекту, которым он будет оперировать. Из этого следует, что код, который изменяет записи в базе данных, следует помещать в триггеры соответствующих полей.
- В отчетах, следует делать четкое различие между логической и визуально обработкой. Это значит, что в триггеры разделов можно помещать код, который управляет внешним видом элементов управления, а также контролирует должен ли отображаться данный раздел или нет. Код же, который оперирует данными, никогда не должен помещаться в триггеры разделов.
- В некоторых случаях, принцип размещения кода поближе к объекту может не соблюдаться. Во-первых, по причинам безопасности. Обычно пользователи не имеют прямого доступа к

DRAFT

таблицам, содержащим критические данные, таким как ***General Ledger Entry***. Если вы поместите код, которые оперирует с таблицей general ledger entry в программный модуль и дадите ему права доступа к таблице, а пользователям право запуска программного модуля, то в результате будет обеспечена безопасность таблицы, а пользователи смогут получать к ней доступ.

- Кроме безопасности есть еще другие причины помещать функции в программный модуль. Функцию, помещенную в программный модуль можно вызвать из многих мест приложения, в том числе тех, о которых вы даже не думали, когда начинали проектировать свое приложение.

Повторное Использование Кода

Возможно, наиболее важная причина, по которой код нужно помещать не ближе к объекту, а отдельно это возможность его повторного использования. Повторное использование кода делает разработку приложений более простой и быстрой. Однако это не главная причина для повторного использования кода. Если вы будете помещать код как предложено, то ваше приложение будет меньше подвержено ошибкам. Централизация кода позволит избежать массы однотипных простых вычислений в разных местах, например, в триггерах элементов управления, которые ссылаются на одно и то же поле таблицы. Если потребуется изменить код, то в такой ситуации можно забыть о некоторых элементах управления или допустить ошибку при редактировании одного из них

17.2 Системные Переменные

C/SIDE автоматически определяет и инициализирует ряд переменных, которые впоследствии можно использовать при разработке приложения. Это системные переменные:

Переменная	Примечание
Rec	Когда запись таблицы (record) изменяется, переменная Rec содержит текущую запись (включая сделанные изменения), в тоже время переменная xRec содержит исходную запись (до изменений)
xRec	
CurrForm	Ссылается на текущую форму. С помощью данной переменной можно получить доступ к элементам управления на форме и изменить динамические свойства, как самой формы, так и ее элементов управления.
CurrReport	Ссылается на текущий отчет, аналогично переменной CurrForm, которая ссылается на текущую форму.
RequestOptionsForm	Ссылается на форму запроса текущего отчета.
CurrFieldNo	Номер текущего поля текущей формы, поддерживается по причинам совместимости.

Кроме того, некоторые триггеры (например, триггер OnFormat) имеют параметры, которые определены как локальные системные переменные.

Пример

Далее приведен пример использования пары записей Rec/xRec.

В приложении данные хранятся в двух таблицах: таблице-заголовке и таблице-строках. В таблице-заголовке содержится общая информация, скажем о заказе продажи, а в таблице-строках содержатся строки продажи. В таблице-заголовке хранится только номер клиента, остальная информация заполняется из таблицы ***Customer*** при выборе значения **Customer No.**.

DRAFT

Можно ли разрешить пользователю изменять номер клиента? В каких-то ситуациях ответ будет Да, в других – Нет. Если заказ уже был отгружен, ответ будет однозначно Нет. Но возможны ситуации, когда ответ должен быть Да, например, чтобы исправить некорректный номер клиента в заказе, при условии, что заказ еще не был обработан.

Это можно сделать следующим образом:

– При выполнении проверки корректности (validate) номера клиента, также нужно проверить был ли заказ отгружен.

– Если заказ был отгружен, то нужно сравнить значение поля с номером клиента в переменных xRec и Rec. Если они различаются, то нужно отменить изменения.

В реальной ситуации, потребуется добавить еще несколько проверок, а также диалоговое окно для пользователя, но основанная идея будет такое же.

17.3 Обработка Ошибок Выполнения

В главе 18, "Отладка C/AL Кода", в разделе Другие Ошибки Выполнения будет описано, как обрабатывать функции возвращающие значение типа Boolean, которое можно обработать, или проигнорировать.

При использовании данных функций, возможно четыре сценария, как показано в следующей таблице:

	Возвращаемое значение игнорируется	Возвращаемое значение обрабатывается
Функция выполнена успешно	Выполнение продолжается	Выполнение продолжается
Произошла ошибка	Возникает ошибка выполнения	Выполнение продолжается. Вы должны обработать ошибку самостоятельно

Типичный пример функции, которая может привести к ошибке выполнения в зависимости от того, как выполняется обработка возвращаемого значения, это функция *GET*. Ее синтаксис:

[Ok :=] Record.GET([Value1], [Value2],...)

Ok переменная типа Boolean, если возвращаемое значение равно TRUE, значит запись найдена, и FALSE, если запись найдена не была.

Если функция GET используется следующим образом:

```
Customer.GET("Customer Number")
```

...и запись не будет найдена, произойдет ошибка выполнения.

С другой стороны, если функция GET используется так:

```
IF Customer.GET("Customer Number") THEN
....
ELSE
...
```

...и запись не будет найдена, выполнение продолжится. В этом случае вы должны обработать ситуацию сами (в части оператора IF, расположенной после ELSE).

DRAFT

17.4 Основные Функции Языка C/AL

Хотя в языке C/AL более 100 функций, постоянно из них используется ограниченный набор, остальные же функции используются от случая к случаю. При разработке простого приложения, возможно, вы будете использовать только 20 различных функций. Это не значит, что остальные функции являются лишними или, что вы ими никогда не воспользуетесь. Однако это означает, что если вы хорошо освоите этот маленький набор функций, то сможете пройти длинную дорогу в программировании на языке C/AL. Когда вам потребуется добавить специализированную функциональность в ваше приложение, вы сможете ближе познакомиться с остальными функциями.

В следующем разделе содержится несколько примеров использования основных функций. Для дополнительной информации по всем функциям следует использовать онлайновую справку *C/SIDE Reference Guide*.

Поиск Записей

Три функции, описанные в данном разделе, используются для поиска записей. Когда вы выполняете поиск записей, важно помнить разницу между функциями GET и FIND, а также как использовать вместе FIND и NEXT.

GET

Функция GET получает одну запись, основываясь на значения полей первичного ключа.

Синтаксис функции GET:

[Ok :=] Record.GET([Value],...)

Например, поле **No.** является первичным ключом таблицы ***Customer***, в этом случае GET можно использовать следующим образом:

```
Customer.GET('4711')
```

В результате из базы данных будет получена запись таблицы customer с номером 4711. Однако функция GET является одной из функций, которые могут привести к ошибке выполнения, если при выполнении произошла ошибка, а возвращаемое значение в коде не обрабатывается. Это значит, что фактически вам следует написать следующий код:

```
IF Customer.GET('4711') THEN
.... // выполняется обработка
ELSE
.... // обработка ошибки
```

Функция GET выполняет поиск записи, не обращая внимания на установленные фильтры, также она не изменяет фильтры. Другими словами, функция GET всегда выполняет поиск по всем записям таблицы.

FIND

Функция Find находит запись в таблице C/SIDE, основываясь на значениях, хранящихся в ключах.

Синтаксис функции Find:

Ok := Record.FIND([Which])

Различия между функциями GET и FIND заключаются в следующем:

- FIND ограничена текущими фильтрами.
- FIND может искать запись, точно совпадающую с указанными значениями ключа, а также большую или меньшую запись.

DRAFT

- FIND может искать первую или последнюю запись (зависит от порядка сортировки текущего ключа).

Эти возможности можно использовать различными способами. При разработке приложений, часто между таблицами используются связи «Один-ко-Многим». Примером, может быть таблица ***Item*** (Товар), в которой один товар может быть связан с несколькими строками таблицы ***Sales Line*** (Строка Продажи). Вы хотите запретить удаление записи в таблице товара, пока существуют открытые заказы продажи, включающие этот товар. Чтобы это проверить, можно использовать функцию FIND.

Посмотрите триггер OnDelete таблицы ***Item***, он включает следующий код:

```
SalesOrderLine.SETCURRENTKEY(Type,"No.");
SalesOrderLine.SETRANGE(Type,SalesOrderLine.Type::Item);
SalesOrderLine.SETRANGE("No.","No.");
IF SalesOrderLine.FIND('-') THEN
  ERROR(Text001,TABLECAPTION,"No.",SalesOrderLine."Document Type");
```

NEXT

Функция NEXT часто используется вместе с функцией FIND, для перехода по записям таблицы.

Синтаксис функции NEXT:

```
Steps := Record.NEXT([Steps])
```

И вместе с функцией FIND:

```
FIND('-');
REPEAT
// обработка записей
UNTIL NEXT = 0;
```

В данном случае функция FIND позиционируется на первой записи таблицы. Затем используется функция NEXT для перехода к следующим записям, пока они не закончатся (тогда функция NEXT вернет 0 (ноль)).

Сортировка и Фильтрация Записей

Следующие функции используются для наложения фильтров на записи таблицы. Т.е. ограничивают возможные значения одного или нескольких полей так чтобы отображалась, модифицировалась, удалялась и т.д. только часть записей. Так же в данном разделе будет рассказано, как изменять порядок сортировки записей в таблице.

SETCURRENTKEY

Данная функция используется для указания текущего ключа для записи и таким образом устанавливает порядок сортировки, используемый таблицей.

Синтаксис функции SETCURRENTKEY:

```
[Ok :=] Record.SETCURRENTKEY(Field1, [Field2],...)
```

При использовании функции SETCURRENTKEY обратите внимание на следующие моменты:

- Неактивные поля игнорируются.
- При поиске подходящего ключа, среда C/SIDE выберет первый ключ, который содержит указанные поля.

DRAFT

Это значит, что:

- Если вы указали только одно поле в качестве параметра для функции SETCURRENTKEY, фактически выбранный ключ может состоять из нескольких полей.
- Если несколько ключей начинается с указанных полей, то выбранный ключ может оказаться отличным от того, который вы ожидали.
- Если не найдено ни одного ключа, начинающегося с указанных полей, то возникнет ошибка выполнения.

  Примечание

  Если вы выполняете проверку возвращаемого функцией значения (Boolean), то ошибка выполнения не произойдет. Однако вы должны определить действия системы, в случае если функция вернет значение FALSE. Иначе выполнение программы продолжится, словно подходящий ключ найден.

SETRANGE

Эта функция используется для установки простых фильтров на поле.

Синтаксис функции SETRANGE:

Record.SETRANGE(Field [,From-Value] [,To-Value]);

В следующем примере:

```
Customer.SETRANGE("No.",'10000','90000')
```

Функция SETRANGE устанавливает на таблицу ***Customer*** фильтр, согласно которому должны быть выбраны только те записи, у которых значения поля **No.** находятся в диапазоне от 10000 до 90000.

При использовании функции SETRANGE нужно помнить:

- SETRANGE удаляет любые фильтры наложенные ранее и заменяет их указанными параметрами From- Value/To-Value.
- Если использовать функцию SETRANGE без параметров From-Value/To-Value, то функция удалит все фильтры наложенные ранее.
- Если указать только параметр From-Value, значение параметра To-Value принимается равным значению From- Value.

SETFILTER

Функция SETFILTER используется для наложения более сложных фильтров.

Синтаксис функции SETFILTER:

Record.SETFILTER(Field, String [, Value],...];

Field – это наименование поля, на которое будет наложен фильтр. String – это выражение-фильтр, которое может содержать %1, %2 и т.д., чтобы указать места, куда система должна подставить переданные в параметрах Value значения.

Ниже приведено два примера:

```
Customer.SETFILTER("No.", '>10000 & <> 20000')
```

Будут выбраны записи, у которых значение поля No. больше 10000 и не равно 20000.

```
Customer.SETFILTER("No.",'>%1&<>%2',Value1, Value2)
```

Если переменным Value1 и Value2 присвоить соответственно значения "10000" и "20000", то результат будет полностью совпадать с предыдущим примером.

DRAFT

GETRANGEMIN

Функция возвращает минимальное значение фильтра, установленного как диапазон значений.

Синтаксис функции GETRANGEMIN:

Record.GETRANGEMIN(Field);

Если текущий фильтр не является диапазоном, то произойдет ошибка выполнения. Например, если установлен следующий фильтр:

```
Customer.SETFILTER("No.",'10000|20000|30000')
```

То при выполнении:

```
BottomValue := Customer.GETRANGEMIN("No.")
```

Произойдет ошибка, т.к. фильтр не является диапазоном.

GETRANGEMAX

Функция GETRANGEMAX работает аналогично функции GETRANGEMIN, за исключением того, что возвращает максимально значение фильтра, установленного как диапазон значений.

Синтаксис функции GETRANGEMAX:

Value := Record.GETRANGEMAX(Field)

Вставка, Изменение и Удаление Записей

Приведенные ниже функции используются для вставки, изменения и удаления записей базы данных.

В целом, все данные функции возвращаются значение типа Boolean, которое показывает, успешно ли произошло выполнение. Если вы не обрабатываете возвращаемое значение в коде, то может произойти ошибка выполнения, если функция вернет значение FALSE. Если обработать возвращаемое значение, проверив его оператором IF, то в этом случает, ошибки не произойдет и можно выполнить корректирующие операции, зная, что выполнение функции не было успешным.

INSERT

Данная функция вставляет запись в таблицу.

Синтаксис функции INSERT:

[Ok :=] Record.INSERT([RunTrigger])

Пример

Создайте следующие переменные:

Variable	Data Type	Subtype
Customer	Record	Customer

```
Customer.INIT;
Customer."No." := '4711';
Customer.Name := 'John Doe';
Customer.INSERT;
```

В результате будет вставлена новая запись, у которой поля No. и Name будут заполнены согласно присвоенным значениям, остальные поля будут заполнены значениями по умолчанию. Если поле

DRAFT

No. является первичным ключом таблицы ***Customer***, то запись будет вставлена в таблицу, только в том случае если в ней нет записи с таким же значением первичного ключа. В этом случае произойдет ошибка выполнения, т.к. в данном примере не выполняется проверка возвращаемого значения.

MODIFY

Данная функция используется для изменения существующей записи таблицы.

Синтаксис функции MODIFY:

[Ok :=] Record.MODIFY([RunTrigger])

Аналогично функции INSERT, возвращает значение Boolean – TRUE, если запись была изменения и FALSE, если запись не существует.

Пример

Создайте следующие переменные:

Variable	Data	Type Subtype
Customer	Record	Customer

```
Customer.GET('4711');
Customer.Name := 'Andrey Panko';
Customer.MODIFY;
```

В результате наименование клиента 4711 будет изменено на Andrey Panko.

MODIFYALL

Функция используется для группового обновления записей.

Синтаксис функции MODIFYALL:

Record.MODIFYALL(Field, NewValue [, RunTrigger])

MODIFYALL ограничена текущими фильтрами, это означает, что будет обновлен только определенный набор записей в таблице. MODIFYALL не возвращает никаких значений.

Пример

Создайте следующие переменные:

Variable	Data Type	Subtype
Customer	Record	Customer

```
Customer.SETRANGE("Salesperson Code",'PS','PS');
Customer.MODIFYALL("Salesperson Code",'JR');
```

С помощью функции SETRANGE будут выбраны все записи, у которых в поле **Salesperson Code** установлено значение PS, а функция MODIFYALL изменяет значение поля **Salesperson Code** выбранных записей на JR.

DELETE

Данная функция используется для удаления записи из базы данных.

Синтаксис функции DELETE:

DRAFT

[Ok :=] Record.DELETE([RunTrigger])

Перед тем как вызывать функцию DELETE нужно определить запись, которую требуется удалить (для этого следует использоваться значения полей, входящих в первичный ключ). Это означает, что функция DELETE не учитывает установленные фильтры.

Пример

Создайте следующие переменные:

Variable	Data Type	Subtype
Customer	Record	Customer

В следующем примере с помощью функции DELETE будет удалена запись с номером 4711:

```
Customer."No." := '4711'
Customer.DELETE
```

Функция DELETE возвращает значение типа Boolean: TRUE если запись удалена и FALSE, если запись не была удалена. Если функция DELETE возвратит значение FALSE, а возвращаемое значение не обрабатывается, то произойдет ошибка выполнения.

При создании собственных приложений, рекомендуется следовать следующим сценариям:

1. Получить запись из базы данных.

2. Выполнить различные проверки, на предмет, можно ли удалить данную запись.

3. Если согласно шагу 2 удаление разрешено, то удалить запись.

В многопользовательском окружении это может привести к проблеме. Так как другой пользователь в это же время может изменить или удалить запись, пока вы выполняете шаги 2 и 3. Если запись изменена, возможно, новое содержание ее полей повлияет на решение об удаление. Также, если запись была удалена другим пользователем, то вы можете получить неожиданную ошибку выполнения, хотя на шаге 1вы уже проверили, что запись существует.

Если при проектировании приложения, вы предполагаете, что можете столкнуться с такой проблемой, то вам следует использовать функцию LOCKTABLE (она будет описана в следующем разделе), но функция LOCKTABLE должна использоваться как можно реже, так как эта функция противоречит концепции оптимистической конкуренции и уменьшает производительность.

DELETEALL

Данная функция используется для удаления всех записей, согласно установленным фильтрам. Если фильтры не установлены, то из таблицы будут удалены все записи.

Синтаксис функции DELETEALL:

Record.DELETEALL([RunTrigger])

Пример

Создайте следующие переменные:

Variable	Data Type	Subtype
Customer	Record	Customer

Из таблицы ***Customer*** будут удалены все записи, в которых в поле **Salesperson Code** установлено значение PS:

```
Customer.SETRANGE("Salesperson Code", 'PS', 'PS')
Customer.DELETEALL
```

DRAFT

Примечание

При использовании Delete(TRUE) в цикле или DELETEALL(TRUE) нет различий в производительности.

Транзакции

Обычно, при разработке приложений в среде C/SIDE вам не нужно заботиться о транзакциях и блокировках таблиц. В главе ГЛАВА 27 СРЕДА C/SIDE В МНОГОПОЛЬЗОВАТЕЛЬСКОМ ОКРУЖЕНИИ будут приведены детальные объяснения. Однако в некоторых ситуациях нужно блокировать таблицы явно. Например, в начале функции вы проверяете данные, содержащиеся в таблице, затем выполняете различные проверки и расчеты с участием этих данных. И в конце, вы хотите вставить запись, основывающуюся на результатах выполненных расчетов. Чтобы данные расчеты имели смысл, нужно чтобы данные, полученные перед началом вычислений, были такими же, как в настоящий момент. Другими словами, вы хотите запретить другим пользователям изменять данные в таблице, пока ваша функция выполняет расчеты.

LOCKTABLE

Решением описанной проблемы является использование функции LOCKTABLE, чтобы заблокировать таблицу при начале работы вашей функции.

Синтаксис функции LOCKTABLE:

Record.LOCKTABLE([Wait] [, VersionCheck])

Дополнительную информацию об одновременной работе в Dynamics NAV, см Непротиворечивость Чтения и Конкуренция (Read Consistency and Concurrency).

Работа с Полями

Следующие функции используются для выполнения операций над полями.

CALCFIELDS

Функция CALCFIELDS используется для обновления вычисляемого поля (FlowFields). Как было описано в разделе Что такое Свойства Формы и Элемента Управления? вычисляемые поля автоматически обновляются, если они являются источником для элемента управления. Однако вычисляемые поля нужно принудительно рассчитывать, если они являются частью более сложного выражения.

Синтаксис функции CALCFIELDS:

[Ok :=] Record.CALCFIELDS(Field1, [Field2],...)

При использовании вычисляемых полей (FlowFields) в функциях языка C/AL, чтобы их обновить нужно, использовать функцию CALCFIELDS. В следующем примере с помощью функции SETRANGE будет установлен фильтр, затем будет вызвана функция CALCFIELDS. Функция CALCFIELDS вычислит поля **Balance** и **Balance Due,** с учетом текущих фильтров.

Пример

Создайте следующие переменные:

Variable	Data Type	Subtype
Customer	Record	Customer

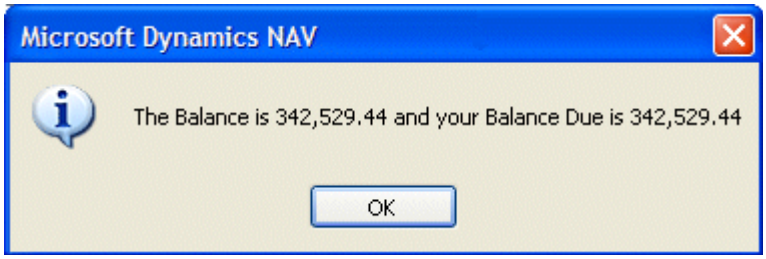
Следующий код рассчитывает значения полей Balance и Balance Due для клиента 01454545:

```
Customer.GET('01454545');
Customer.SETRANGE("Date Filter",0D,TODAY);
```

Customer.CALCFIELDS(Balance,"Balance Due");

MESSAGE('The Balance is %1 and your Balance Due is %2',Customer.Balance,Customer."Balance Due");

И показывает следующее сообщение:



CALCSUMS

Функция CALCSUMS используется для расчета сумм для одного и более полей, которые входят в SumIndexFields.

Синтаксис Функции CALCSUMS:

[Ok :=] Record.CALCSUMS (Field1, [Field2],...)

Для работы функции CALCSUMS нужно выбрать ключ, который содержит соответствующие поля SumIndexField. Как CALCFIELDS, так и CALCSUMS, при выполнении расчетом принимают во внимание установленные фильтры.

Пример

Создайте следующие переменные:

Variable	Data Type	Subtype
Custledgerentry	Record	Cust.Ledger Entry

В данном примере выбирается соответствующий ключ, устанавливаются некоторые фильтры, выполняется расчет и в конце отображается сообщение.

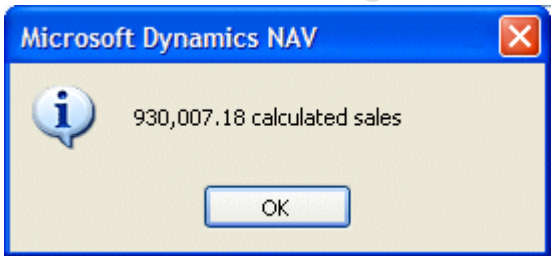
Custledgerentry.SETCURRENTKEY("Customer No.");

Custledgerentry.SETRANGE("Customer No.",'10000','50000');

Custledgerentry.SETRANGE("Posting Date",0D,TODAY);

Custledgerentry.CALCSUMS("Sales (LCY)");

MESSAGE ('%1 calculated sales', Custledgerentry."Sales (LCY)")



FIELDERROR

Функция FIELDERROR вызывает ошибку выполнения и отображает сообщение об ошибке связанное с полем.

DRAFT

Синтаксис функции FIELDERROR:

Record.FIELDERROR(Field, [Text])

Данная функция очень похожа на функцию ERROR, описанную ранее. Однако в ней есть несколько полезных особенностей. Во-первых, она проще в использовании. Но более важной причиной для применения этой функции является то, что если наименование поля изменилось в **Table Designer** (например, переведено на другой язык), то сообщение выводимое функцией FIELDERROR будет содержать текущее наименование поля.

Функцию FIELDERROR легко вызвать:

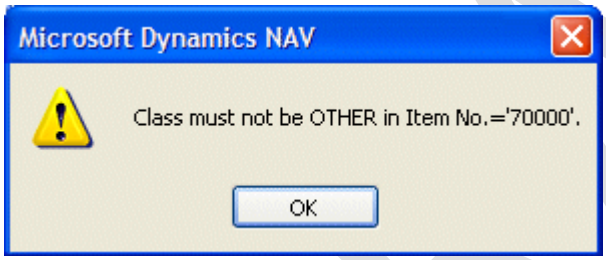
Пример

Создайте следующие переменные:

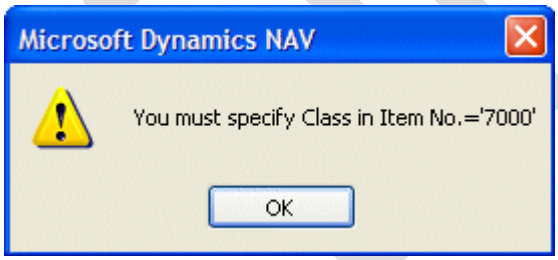
Variable	Data Type	Subtype
Item	Record	Item

```
Item.GET('70000');
IF Item.Class <> 'HARDWARE' THEN
  Item.FIELDERROR(Class);
```

Если в поле Class товара 70000 будет содержаться значение отличное от «hardware», появится следующее сообщение:



Если поле с типом text или code содержит пустую строку, то будет отображено следующее сообщение:



Пустое числовое поле, интерпретируется как поле, содержащее значение 0 (ноль), в этом случае сообщение об ошибке будет таким же, как на первой картинке, но вместо "OTHER" будет отображен "0".

И наконец, если текст отображаемый системой по умолчанию не подходит для вашего приложения, то можно добавить свой текст. В этом случае функция FIELDERROR вызывается следующим образом:

Пример

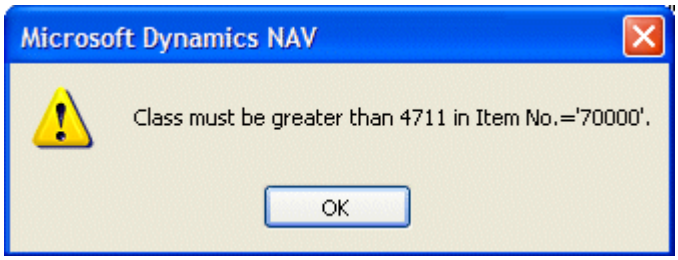
Создайте следующие переменные:

Variable	Data Type
Class	Code

DRAFT

```
IF Item.Class < '4711' THEN
  Item.FIELDERROR(Class,'must be greater than 4711');
```

Будет показано следующее сообщение об ошибке:



FIELDNAME

Функция FIELDNAME возвращает наименование поля.

Синтаксис функции FIELDNAME:

Name := Record.FIELDNAME(Field)

Использование функции FIELDNAME позволяет создавать сообщения, в которых всегда содержится наименование поля, даже если оно было изменено. Также функцию FIELDNAME можно использовать вместе с функцией FIELDERROR:

FIELDERROR (Quantity,'must not be less than ' + FIELDNAME ("Quantity Shipped"));

INIT

Функция INIT выполняет инициализацию записи.

Синтаксис функции INIT:

Name := Record.FIELDNAME(Field)

Если для полей записи были заданы значения по умолчанию (свойство *InitValue*), эти значения будут использованы при инициализации, и переопределят значения, используемые согласно типам данных полей.

Функция INIT *не выполняет* инициализацию полей первичного ключа.

TESTFIELD

Данная функция используется, чтобы проверить, содержит ли поле определенное значение.

Синтаксис функции TESTFIELD:

Record.TESTFIELD(Field, [Value])

Если проверка покажет, что поле не содержит определенное значения, то функция вызовет ошибку выполнении и отобразит сообщение об ошибке. Это значит, что все изменения, выполненные для записи, не будут учитываться. Если выполняется проверка на пустое значение, то значение поля не должно быть пустым или быть равным 0 (ноль).

Пример

Создайте следующие переменные:

Variable	Data Type	Subtype
customer	Record	Customer

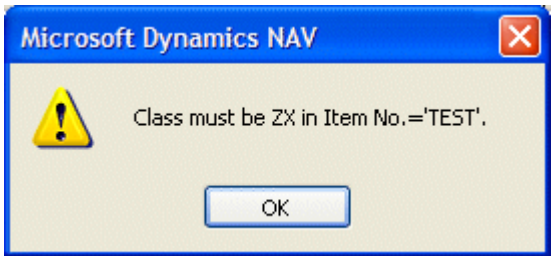
Следующий код:

DRAFT

customer.GET('10000')

customer.TESTFIELD("Language Code",'ZX');

Проверяет, чтобы поле **Language Code** для клиента 10000 в таблице ***Customer*** было равно ZX. Если значение поля отличается, то будет отображено следующее сообщение:



VALIDATE

Функция VALIDATE используется для вызова триггера OnValidate для поля.

Синтаксис функции VALIDATE:

Record.VALIDATE(Field [, NewValue])

Пример

При вводе номера финансового счета в журнал, система выполняет код в триггере таблицы и переносит в журнал значение из поля name счета.

Если ввод номера счета выполняется с помощью пакетного задания, система автоматически не будет выполнять код, по переносу наименования счета. С помощью следующего кода можно указать системе, что необходимо выполнить код из триггера поля.

Создайте следующие переменные:

Variable	Data Type	Subtype
GeneralLedgerEntry	Record	G/L Entry

GenJnlLine.VALIDATE("G/L AccountNo", '100')

Или

GenJnlLine."G/L AccountNo" := '100'

GenJnlLine.VALIDATE("G/L AccountNo")

Функция VALIDATE полезна при централизации обработки и упрощения приложения в части поддержки.

Например, если значение поля **Total Amount** рассчитывается с использованием значений трех других полей, то расчет должен выполняться при изменении любого из этих полей. В этом случае следует избежать ввода формул расчета в триггеры OnValidate каждого поля, т.к. при изменении формулы расчета, нужно будет ввести изменения во все триггеры, что может привести к ошибке. Вместо этого следует ввести формулу расчета в триггер OnValidate только одного из этих полей, и вызывать этот триггер из триггеров OnValidate других полей.

Сообщения и Диалоговые Окна

Для отображения сообщений и ввода информации существует несколько специализированных функций. Однако чтобы гарантировать, что ваше приложение имеет единообразный интерфейс, следует использовать формы. Тем не менее, существует ряд ситуаций, когда оправдано использование функций по работе с диалоговыми окнами. Главное предназначение функций по работе с диалоговыми окнами заключается:

DRAFT

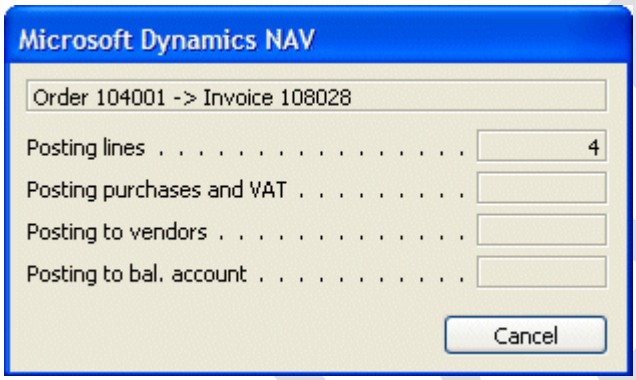
- Отображать окно с информацией о ходе выполнения, какого-либо длительного процесса.
- Остановить выполнение программы и отобразить сообщение об ошибке
- Дать возможность пользователю подтвердить выбор, перед тем как программа продолжит выполнение.

Кроме этого, функция STRMENU предоставляет удобный способ создания форм, для отображения пользователю набора опций. Это намного быстрее, чем создавать отдельную форму, для предоставления ограниченного набора опций.

Создание Окна для Отображения Хода Выполнения

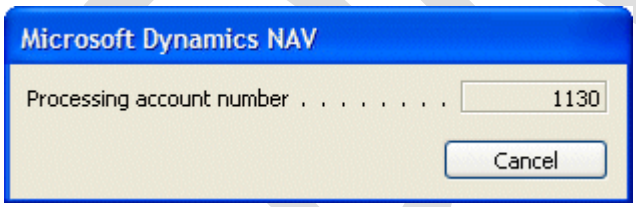
Если приложение выполняет какие-либо операции значительное время, следует отображать окно, которое будет информировать пользователя о ходе выполнения. Данная информация не нужна для работы программы, однако она показывает пользователю, что приложение работает.

Автоматически в диалоговое окно добавляется кнопка Cancel, которая позволяет пользователю остановить выполнение. В некоторых случаях можно создать элемент управления индикатор, как это сделать описано в разделе «Использование Индикаторов для Отображения Значений», в других случая можно создать подобное окно:



В этом окне, каждое поле обновляется в процессе работы программы. В данном примере, поля используются для подсчета выполненных учетных операций.

В других случаях, можно отображать номер счета, который обрабатывается в данный момент:



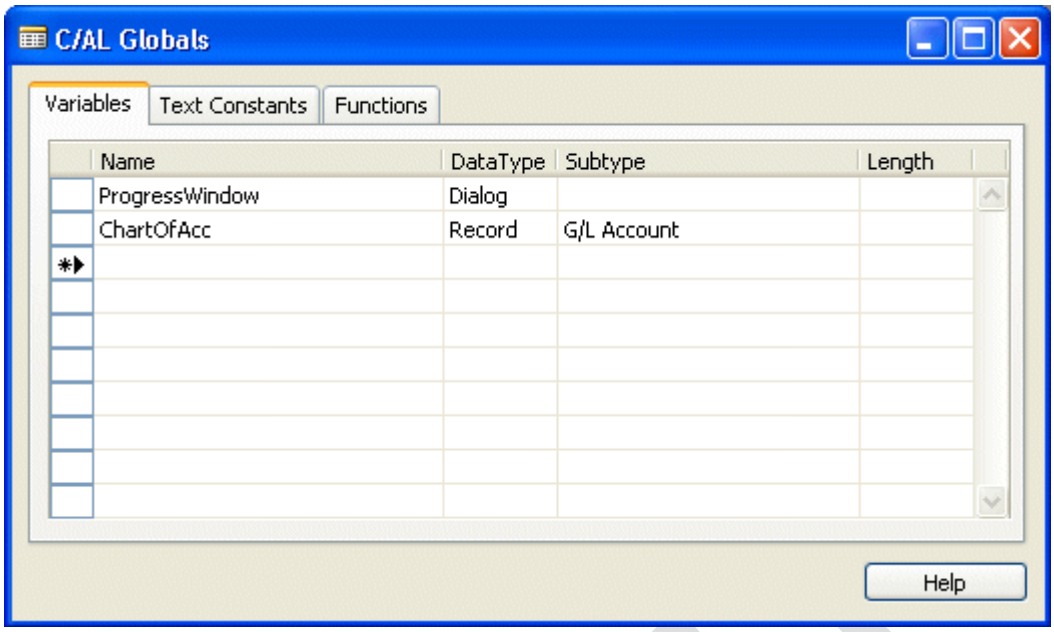
Чтобы создать такое окно:

1. Откройте Object Designer и создайте новый программный модуль.

2. Создайте переменную ProgressWindow с типом данных Dialog.

3. Создайте переменную ChartOfAcc с типом Record и подтипом G/L Account.

DRAFT



4. Откройте окно ***C/AL Editor*** и добавьте следующий код:

```
ProgressWindow.OPEN('Processing account number #1#######');
REPEAT
  SLEEP(1000);
  ProgressWindow.UPDATE(1,ChartOfAcc."No.");
  // обработка счета...
UNTIL ChartOfAcc.NEXT = 0;
ProgressWindow.CLOSE;
```

Первая строка определяет строку, которая будет отображаться в окне. Часть строки с символами решетки (#) и номером, определяет поле, которое будет отображаться в окне. Число (1) относится к полю.

В данном примере, при обновлении каждой записи таблицы ***G/L Account*** отображается номер счета.

Функция SLEEP(1000); совершенно не нужна и используется только, чтобы замедлить процесс обработки, и дать возможность рассмотреть окно.

Прочие Сообщения для Пользователя

Также существуют функции по работе с диалоговыми окнами, которые позволяют отображать пользователю короткие сообщения. Все эти функции кроме MESSAGE, останавливают выполнение программы, пока не будет получен ответ от пользователя.

MESSAGE

Функция MESSAGE отображает пользователю сообщение, которое будет показано на экране до тех пор, пока пользователь не щелкнет кнопку OK.

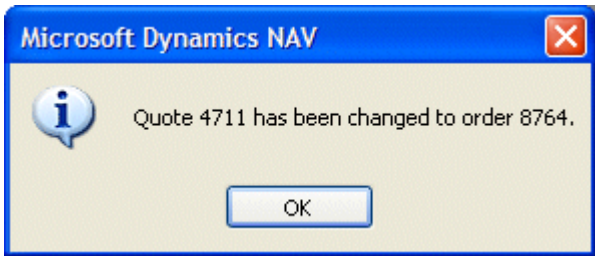
Синтаксис функции MESSAGE:

MESSAGE(String [, Value1,...])

Функция MESSAGE выполняется асинхронно, это значит, что сообщение не будет показано пользователю до тех пор, пока функция, из которой была вызвана функция MESSAGE, не завершит

DRAFT

работу или другая функция потребует ввода информации от пользователя. Функция полезна для информирования пользователя об успешном окончании процесса:



В качестве примера можно посмотреть программный модуль 83. Триггер OnRun, кроме вызова кода по преобразованию Квоты (Quote) в Заказ Продажи (Sales Order) заканчивается выводом сообщения. Сообщение генерируется следующим кодом:

```
MESSAGE(Text001,"No.",SalesHeader2."No.")
```

Text001 это текстовая константа, которая содержит следующий текст:

- *ENU:Quote %1 has been changed to order %2.*
- *RUS: Квота %1 преобразована в заказ %2.*

Примечание

В отличие от окна, отображающего ход выполнения, для использования функции MESSAGE не нужно вначале объявлять переменную типа Dialog. Функция MESSAGE создает свое собственное окно.

ERROR

Функция ERROR очень похожа на функцию MESSAGE за исключением одной детали: когда пользователь подтверждает сообщение, выполнение завершается. См. Также описание функции FIELDERROR.

Синтаксис функции ERROR:

ERROR(String [, Value1,...])

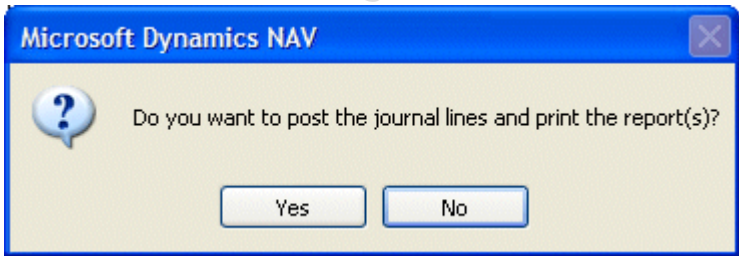
CONFIRM

Функция CONFIRM используется аналогично функции MESSAGE для отображения сообщений.

Синтаксис функции CONFIRM:

Ok: = Dialog.CONFIRM(String [, Default] [, Value1],...)

В отличие от функции MESSAGE, функция CONFIRM возвращает значение, которое можно (и нужно) использовать. Возвращаемое значение зависит от кнопки, нажатой пользователем в ответ на вопрос:



Окно создается с помощью следующего кода:

```
IF CONFIRM ('Do you want to post the journal lines and print the report(s)?',FALSE) THEN
  Message ('Posting')
```

DRAFT

```
ELSE
  Message ('No Posting');
EXIT;
```

Параметр FALSE означает, что кнопкой по умолчанию будет кнопка No.

Форма Быстрого Выбора Опций

Функция STRMENU используется для создания и отображения пользователю формы с группой опций, а также возвращает выбранное пользователем значение.

STRMENU

Синтаксис функции STRMENU:

OptionNumber := Dialog.STRMENU(OptionString [, DefaultNumber]);

OptionNumber – это номер опции, которую выбрал пользователь. Первая опция в OptionString идет под номером 1. Если пользователь закроет форму с помощью клавиши ESC, STRMENU вернет 0 (ноль). Если указано DefaultNumber, то выделяется опция, используемая по умолчанию. Если параметр DefaultNumber не определен, то система по умолчанию выделяет первую опцию.

Пример

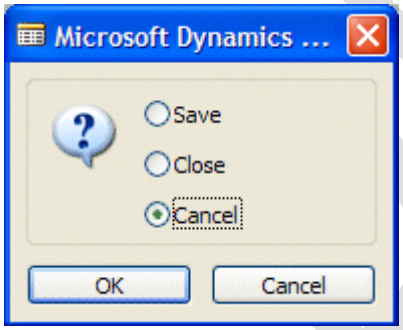
Создайте следующие переменные:

Variable	Data Type
Selection	Integer

Следующий код:

```
Selection := Dialog.STRMENU('Save,Close,Cancel',3)
```

Создает такое окно:



По умолчанию выбрана опция Cancel потому, что в качестве параметра DefaultNumber передано значение 3. Это правильно использовать в качестве опции по умолчанию «безвредное» действие, типа Cancel (Отмена), т.к. эту опцию можно выбрать с помощью клавиши ENTER. Если пользователь случайно нажмет клавишу ENTER, никакие непоправимые действия не произойдут, как могло бы случиться с опцией "Delete all" (Удалить Все).

DRAFT

ГЛАВА 18 ОТЛАДКА C/AL КОДА

В данной главе описана природа программных ошибок, а также рассказано как использовать Dynamics NAV Debugger для их поиска.

- Что такое программная ошибка?
- Microsoft Dynamics NAV Debugger
- Инструмент Код Приложения

18.1 Что Такое Программная Ошибка?

При разработке приложений с использование C/AL кода встречаются три вида ошибок:

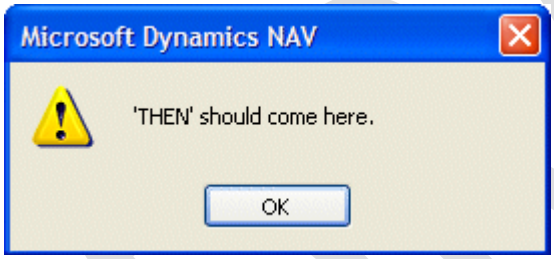
- Синтаксические ошибки (Syntax errors)
- Ошибки выполнения (Run-time errors)
- Ошибки в логике программы (Program logic errors)

Процесс поиска и исправления программных ошибок называется *Отладка*.

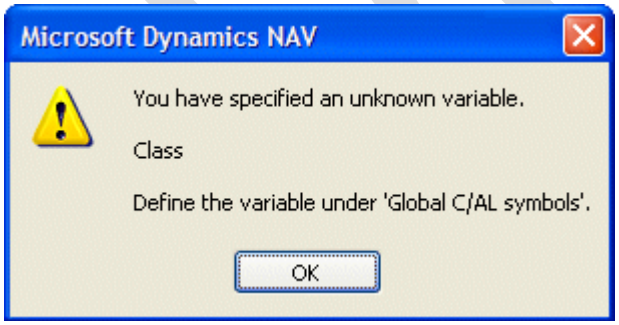
В данной главе будет рассказано, как искать и выявлять ошибки, а также показано, как использовать Dynamics NAV Debugger (отладчик) для поиска ошибок выполнения и ошибок в логике программы.

Синтаксические Ошибки (Syntax Errors)

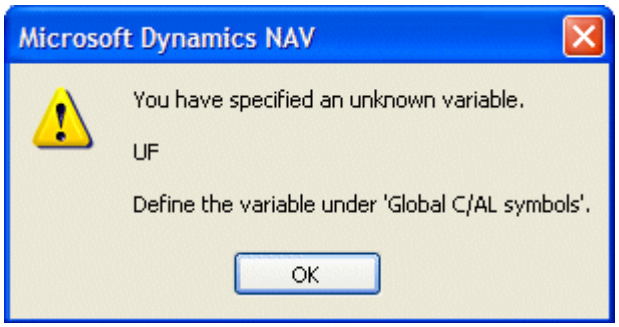
Эти ошибки выявляет компилятор C/AL, при попытке компилировать C/AL, независимо от того находится код в программном модуле или в другом объекте (таблице, форме, отчете, датапорте или программном модуле). При обнаружении ошибки, компилятор уведомит вас таким сообщением:



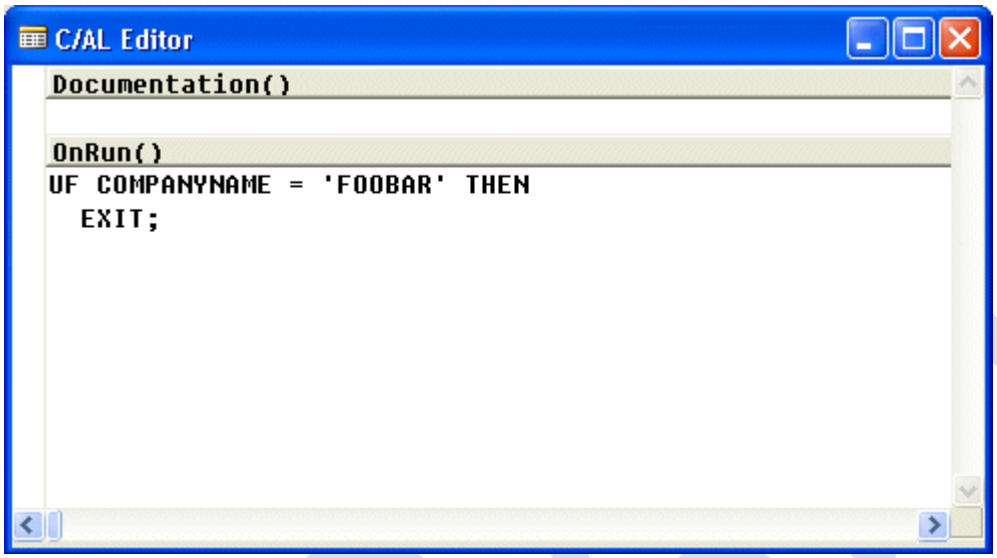
Или таким:



Если нажать ENTER, то появится C/AL editor с курсором, установленным перед проблемным оператором. Обратите внимание, что сообщение об ошибке не всегда отражает природу ошибки. Рассмотрим следующее сообщение:



Если посмотреть на проблемный код в редакторе, то станет ясно, что с неизвестной переменной не нужно ничего делать:



Ошибка в данном случае заключалась в неправильном написании оператора IF, который был введен как UF. С точки зрения компилятора UF это неизвестный идентификатор, отсюда такое сообщение об ошибке. Если посмотреть на код, то легко увидеть, в чем заключается проблема на самом деле.

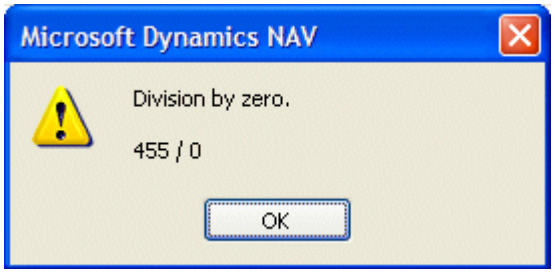
Компилятор не будет компилировать код, который содержит какие-либо синтаксические ошибки, как пропуск THEN в операторе IF или использование необъявленных переменных.

Ошибки Выполнение (Run-time Errors)

Ошибки выполнения происходят во время выполнения программы. Эти ошибки не определяются компилятором, так как с точки зрения синтаксиса они корректны. Хорошим примером является деление на ноль. Представим следующий код:

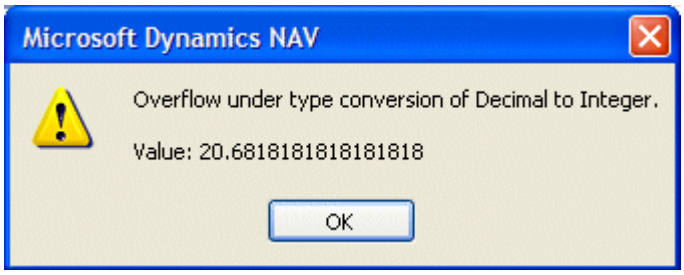
Ratio := First_number / Second_number;

Здесь нет синтаксических ошибок, но данный код может привести к появлению следующей ошибки:



Эта произойдет, если переменной Second_number будет присвоено значение 0 (ноль), что приведет к делению на ноль.

Если все три переменные будут принадлежать к типу данных integer, то возможно появление такой ошибки:



Эта ошибка произойдет потому, что результат деления не может быть присвоен переменной с типом integer. В процессе выполнения операции результат был преобразован в тип decimal, но при преобразовании обратно в integer (присвоение переменной Ratio) возникла ошибка.

Эти ошибки объединяет то, что во многих ситуациях код работает хорошо, но в некоторых случаях возникает ошибка. Основная опасность данных ошибок заключается в том, что хотя с точки зрения синтаксиса в коде все в порядке, ошибка может произойти, когда программа находится в эксплуатации.

> **Об ошибках выполнения и целостности базы данных**
>
> Не нарушают ли ошибки выполнения целостность данных. Например, если в триггере были обновления значения в некоторых полях, а затем произошла ошибка выполнения, до того как были обновлены оставшиеся поля. В главе ГЛАВА 27 СРЕДА C/SIDE В МНОГОПОЛЬЗОВАТЕЛЬСКОМ ОКРУЖЕНИИ будет объяснено, как поддерживается целостность данных. При начале работы триггера, начинается транзакция записи. Если в триггере возникнет ошибка выполнения, транзакция записи будет отменена, а выполнения триггера остановлено.

Как Избежать Ошибок Выполнения

В целом, ошибки выполнения не должны происходить. Так и будет, если внимательно относится к процессу программирования. Ниже будет приведено несколько рекомендаций о том, как избежать ошибок выполнения. Но это только рекомендации, так как возникновение ошибок выполнения сильно зависит от остального кода в вашем приложении. С другой стороны, если вы уверены, что этот другой код полностью исключает возникновение ошибки, то вы можете не обрабатывать ее дополнительно (например, перед тем как вызываться функцию GET выполнялась проверки, что запись существует).

Существует две категории ошибок выполнения: ошибки связанные с использованием типов данных и ошибки при выполнении функций. Деление на ноль не подходит ни к одной из этих категорий, но оно будет отнесено в первой категории.

Заголовок данного раздела слишком оптимистичен, на самом деле вы можете избежать только часть ошибок (в основном, связанный с использованием типов данных). Остальные ошибке не удастся избежать полностью, но можно написать код, который защитит от ошибок пользователя. Этот код заменит стандартную обработку ошибок (которая состоит из отображения сообщения об ошибке, закрытия формы, которая была активна в момент возникновения ошибки и отмены всех изменений в базе данных). Вы можете написать код обработки ошибок лучше, например, дать пользователю шанс исправить значение, которое вызвало ошибку, или как минимум отобразить сообщение с детальной информацией о причинах ошибки.

Ошибки, Связанные с Типами Данных

Самый простой способ избежать этих ошибок – использовать правильные типы данных. Ошибок, связанных с преобразованием типов, или ошибок переполнения можно избежать, если использовать соответствующие типы данных. Если вернуться, к примеру, приведенному выше, то можно сказать, что использование типа данных integer было не лучшим вариантом для переменной

DRAFT

Ratio. См. ГЛАВА 16 ВВЕДЕНИЕ В ЯЗЫК C/AL для информации о типах данных и ГЛАВА 25 ПРЕОБРАЗОВАНИЕ ТИПОВ ДАННЫХ, чтобы узнать, как выполняется процесс преобразования типов данных в среде C/SIDE.

Ошибки деления на ноль можно избежать несколькими способами, в зависимости от контекста, где используется данный фрагмент. Если пользователь вводит делитель (переменная Second_number) в текстовую зону, то перед тем как рассчитывать результат, можно выполнить проверку значения переменной Second_number и отказаться от расчета, если ее значение 0 (ноль):

```
IF Second_number <> 0 THEN
  Ratio: = First_number / Second_number
ELSE
  MESSAGE ('Переменная Second_number не должна быть равна 0');
```

Если Second_number это поле в таблице базы данных, и в нем никогда не должно быть значение 0 (ноль), то данную проверку лучше всего выполнять в триггере поля OnValidate. В этом случае, вы будете уверены, что в поле нельзя ввести значение 0 (ноль), независимо от того сколько различных форм и текстовых зон используется для ввода значений в данное поле.

Другие Ошибки Выполнения

Любая функция может вызвать ошибку выполнения, если ей не удалось выполнить то, для чего она предназначена. Хорошим примером является функция GET, которая находит запись в базе данных согласно полям первичного ключа.

Синтаксис функции GET:

[Ok: =] Record.GET([Value1], [Value2],...)

Возвращаемое значение Ok является типом Boolean. Если запись найдена, то возвращается значение TRUE, иначе - FALSE. Возвращаемое значение можно проигнорировать, на это указывают квадратные скобки. Если проигнорировать возвращаемое значение, а требуемая запись не будет найдена, то возникнет ошибка выполнения и будет отображено сообщение, созданное системой. Если же обрабатывать возвращаемое значение, то ошибка выполнения не произойдет, это предполагает, что вы самостоятельно обработаете возникшую ситуацию.

В онлайновой справке *C/SIDE Reference Guide* указано, должна ли для других функций выполняться аналогичная обработка. Также можно посмотреть на синтаксис функции в Symbol Menu: возвращает ли функция значение Ok. Если это так нужно обратиться к онлайновой справке *C/SIDE Reference Guide*, так как существует ряд функций, которые возвращают значение типа Boolean для других целей, нежели описанные в данном разделе. Например, функция ASCENDING используется для проверки порядка сортировки таблицы, если возвращаемое значение TRUE сортировка выполнена по возрастанию, FALSE – по убыванию.

Пример

Возвращаемое значение используется в следующих конструкциях:

```
IF NOT Customer.GET("No.") THEN
  Customer.INIT;
```

Или таких

```
IF NOT Customer.GET("No.") THEN BEGIN
  MESSAGE ('Customer %1 not found', "No.");
  EXIT;
END;
```

DRAFT

Таким образом, можно оградить пользователя от ошибок выполнения. В первом примере, если нельзя получить запись с заданным номером из таблицы Customer, то инициализируется новая запись. Во втором примере, пользователь получает уведомление, что запись невозможно получить в базе данных и выполнение триггера завершается. Это всего лишь общие рекомендации, вы должны решать как будут обрабатываться такие ситуации исходя из кода в вашем приложении.

Поиск и Исправление Ошибок Выполнения

Как можно было видеть, сообщения об ошибках ориентированы на конечного пользователя. Поэтому они не содержат ссылку на переменную или функцию, а вместо этого содержат пояснение и «фактическое» значение, которое вызвало ошибку. Это значит, что такие ошибки сложнее находить, чем ошибки синтаксиса.

Чтобы выявить ошибку выполнения, нужно точное описание последовательности событий, которая привела к возникновению ошибки. Т.е. что делал пользователь, когда возникла ошибка, какие значения он вводил, и какая запись при этом обрабатывалась.

Если ошибка была вызвана простой причиной, например, формулой расчета, в которой не была выполнена проверка деления на ноль, то вы легко сможете найти соответствующий код. С другой стороны, обстоятельства которые привели к ошибке, могут быть более сложными, и в этом случае вы не сможете точно определить место ее возникновения, в этом случае можно использовать отладчик (будет описано далее).

Ошибки в Логике Программы

Третьей большой категорией ошибок являются ошибки в логике программы. В данном случае программа компилируется и работает без появления ошибок выполнения, но не выполняет те функции, которые должна.

Можно утверждать, что большинство ошибок выполнения, также являются ошибками в логике программы. Однако «настоящие» ошибки в логике программы, не проявляются как предыдущие ошибки, а вместо этого незаметно генерируют некорректные данные, и их не всегда можно выявить стандартными способами.

18.2 Microsoft Dynamics NAV Debugger

Dynamics NAV включает отладчик, который помогает проверять, исправлять и модифицировать код так, чтобы ваше приложение беспрепятственно работало и выполняло то, что от него ожидается.

Общее Описание

Базовая концепция отладки базируется на Точках Останова (*breakpoint)*, которыми отмечаются программные операторы. Когда ход выполнения программы достигнет такого оператора, то вмешается отладчик и отложит выполнение (breaks) пока вы не укажете ему, что можно продолжать. Без точек останова код выполняется как обычно, даже при запущенно отладчике. Статус Отключенная Точка Останова (*disabled breakpoint)* означает, что точка останова установлена для программного оператора, но в данный момент она отключена (процесс выполнения не будет остановлен).

Если вам нужно выявить ошибку выполнения, то вы просто отключаете в отладчике настройку Break on Triggers и щелкаете кнопку Go. Отладчик автоматически остановит выполнения кода, когда повстречает ошибку.

Также можно использовать отладчик для поиска логических ошибок. Однако поиск логических ошибок может быть не таким простым, и требовать глубокого понимания того, как код должен работать. Отладчик позволяет вам выполнять по одному программному выражению, при этом на каждом шаге вы можете изучать содержимое глобальных переменные, локальных переменных и

DRAFT

текстовых констант. Таким способом, вы можете определить значения, отличающиеся от значений предполагаемых к использованию согласно проекту приложения.

При первом запуске отладчика, по умолчанию включена настройка Точка Останова на Триггерах (SHIFT+CTRL+F12). Хотя код будет исполняться как обычно, т.к. не заданы точки останова, отладчик все равно прервет выполнение, когда достигнет первого триггера. В этот момент можно установить другие точки останова и отключить настройку Точка Останова на Триггерах. Если не отключить данную настройку отладчик будет прерывать выполнение кода на каждом триггере.

Функциональность Выполнение Кода, которая будет описана дальше, позволяет вести протокол, а затем просматривать код, который выполнялся в одной или нескольких транзакциях. Эту функциональность можно использоваться как альтернативу отладчику или использовать их вместе.

Активация Отладчика

Отладчик можно активировать из Dynamics NAV и из NAV Application Server:

Из Dynamics NAV

Чтобы активировать отладчик из Dynamics NAV, щелкните Сервис, Debugger, Active (SHIFT+CTRL+F11). Также можно запустить Dynamics NAV с активным отладчиком из командной строки, для этого нужно использовать параметр debug:

Пример

fin.exe debug

Из NAV Application Server

Чтобы активировать отладчик из NAV Application Server нужно использовать параметр debug:

Пример

nas debug,startupparameter="test",servername=PC0123

Если вы деактивируете отладчик, вы не сможете его повторно активировать, пока не выключите NAV Application Server и не запустите его повторно с параметром debug.

Чтобы иметь возможность активировать отладчик, в папке, куда установлен NAV Application Server, должна находиться лицензия разработчика.

Интерфейс Отладчика

Интерфейс отладчика включает специальные меню, окна и диалоговые окна.

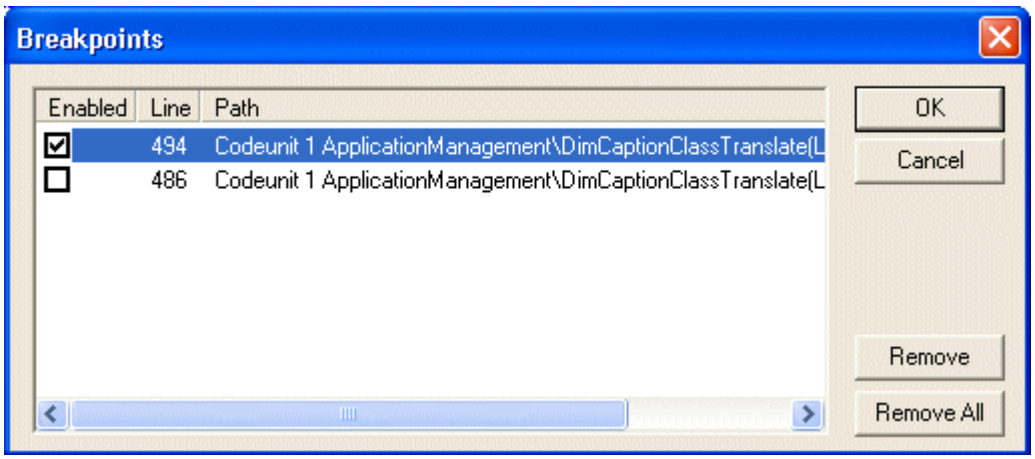
Меню Отладчика

Команды отладчика находятся в меню Edit, View и Debug:

Меню Edit

Из данного меню можно получить доступ к окну ***Breakpoints*** (SHIFT+F9). В нем отображается список всех установленных точек останова (breakpoints). Вы можете включить, отключить и удалить точки останова из списка.

DRAFT



Меню View

Это меню содержит команды, позволяющие отображать различные окна отладчика, такие как окно ***Variables*** или окно ***Call Stack***. Также оно содержит команды для задания размера используемого текста, и команды включения/отключения панелей инструментов standard и debug.

Меню Debug

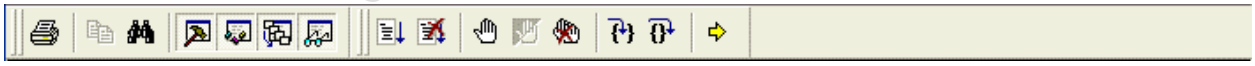
Данное меню содержит команды, которые запускают и управляют процессом отладки, например, Go, Step Into, Step Over и Show Next Statement.

- *Go* выполняет код с текущего программного оператора до точки останова, или пока код не будет завершен, или пока приложение не остановится для выполнения пользовательского ввода.
- *Step Into* выполняет за раз один программный оператор. При выполнении будет выполняться вход во все вызываемые функции. Это значит, что отладка выполняется пошагово.
- *Step Over* выполняет за раз один программный оператор, как и *Step Into*, но если выполнить эту команду для функции, то функция будет выполнена полностью, без пошаговой отладки. Если вы используете данную команду с настройкой Точка Останова на Триггерах, отладчик будет прерывать выполнение на каждом триггере. Кроме того, если внутри функции, для которой использовалась команда step over, есть точка останова, то отладчик остановится на точке останова.
- *Show Next Statement* показывает следующий программный оператор в коде.

Также меню Debug содержит команды для установки, включения и отключения точек останова. Обратите внимание, что настройка Точка Останова на Триггерах не связана с другими точками останова, поэтому на нее не влияет команда Remove All Breakpoints.

Панель Инструментов Debugger

Кнопки на панели инструментов соответствую командам, доступным в меню.



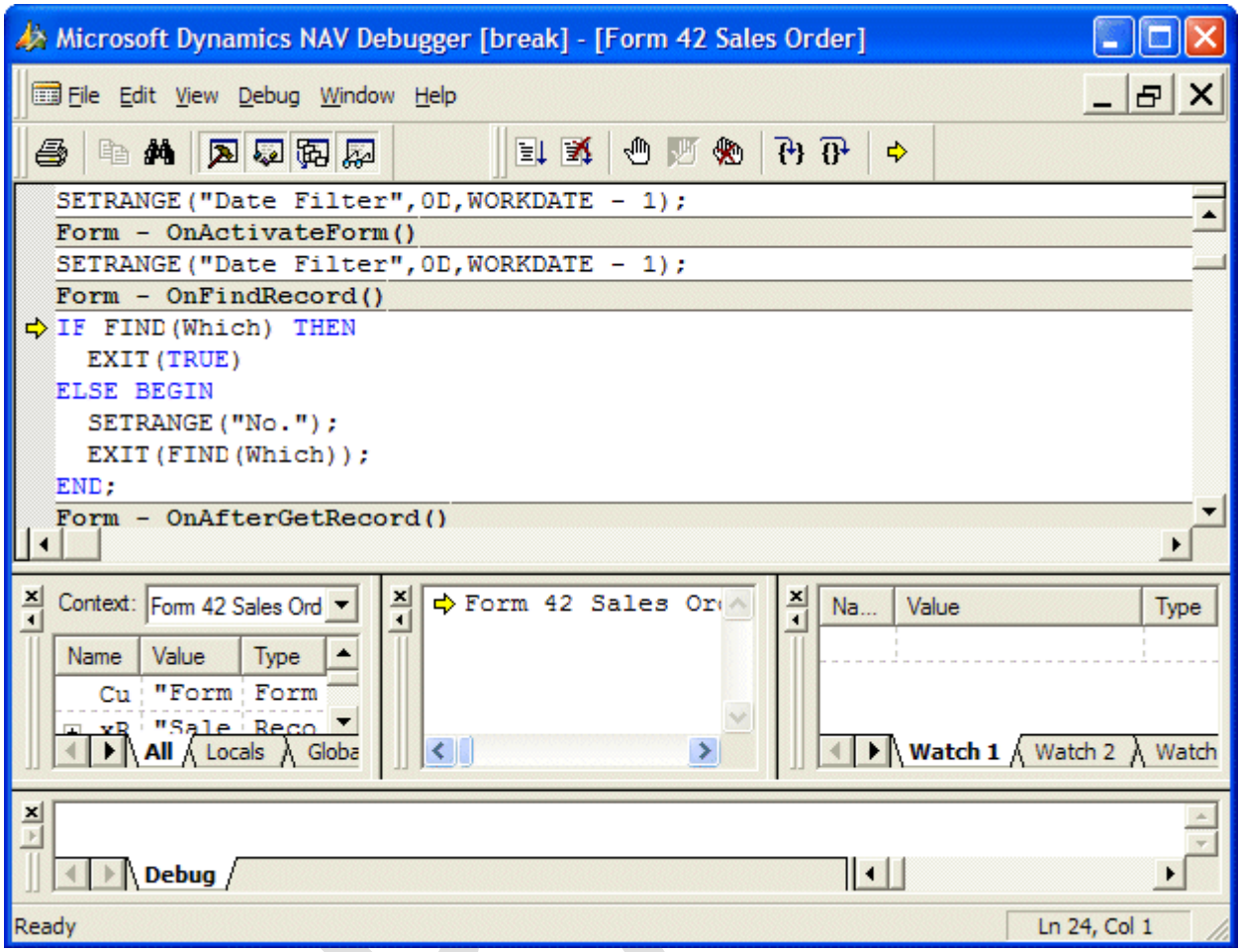
Используйте данную панель инструментов, чтобы:

- Скрыть/отобразить окна ***Output***, ***Variables***, ***Call Stack*** и ***Watch***.
- Вставить/Удалить Точку Останова
- Включить/Отключить Точку Останова
- Удалить все Точки Останова
- Кнопки Go и Stop Debugging, Step Into и Step Over, Show Next Statement

DRAFT

Окно Отладчика

Для отображения отладочной информации используется четыре окна: ***Output***, ***Variables***, ***Call Stack*** и ***Watch***. К этим окнам можно получить доступ из меню View и из панели инструментов Standard.



Окно *Output.* Отображает информацию, связанную с процессом отладки.

Окно *Variables.* Показывается названия переменных и их значения, а также отмечает переменные, значение которых изменилось. В окне доступны четыре закладки: ***All***, ***Locals***, ***Globals*** и ***Text Constants***. Вы не можете добавить переменные в окно ***Variables*** (для этого нужно использовать окно ***Watch***). Вы можете развернуть или свернуть отображение переменной в виде дерева. Вы можете развернуть переменную, если в поле **Name** есть символ (+). Если там отображается (–), то переменная уже развернута.

Окно *Call Stack.* Отображает стек вызовов для активной функции. Отладчик отображает активную функцию вверху стека, а более старые вызовы под ней. Если дважды щелкнуть по линии стека вызовов, то слева от строки появится зеленая стрелка. А в окне, содержащем отлаживаемый код, зеленая стрелка покажет место, где находится триггер, в котором был выполнен вызов соответствующий выбранной строке стека.

Окно *Watch.* Окно ***Watch*** используется для отслеживания значений переменных, представляющих особый интерес, при отладке программ. В это окно можно перетащить название переменной, которую нужно отслеживать, из окна ***Variables*** или из окна, содержащего отлаживаемый код. Также можно ввести наименование переменной непосредственно в данное окно.

Окно ***Watch*** содержит три вкладки: ***Watch1***, ***Watch2*** и ***Watch3***. С их помощью можно группировать переменные, которые нужно отслеживать вместе. Например, переменные, связанные с определенной формой, можно поместить на одну закладку, а переменные, связанные с диалоговым окном, на другую. Таким образом, можно смотреть переменные одной закладки при отладке формы, а переменные второй закладки при отладке диалогового окна.

Символы, Используемые в Интерфейсе Отладчика

В интерфейсе отладчика используются следующие символы:

Символ	Значение
●	С данным программным оператором связана активная точка останова
○	Точка останова выключена
➪	Данный программный оператор будет выполнен
▶	Указывает на строку стека, по которой был выполнен двойной щелчок. Также эта зеленая стрелка покажет место, где находится триггер, в котором был выполнен вызов соответствующий выбранной строке стека (в окне, содержащем отлаживаемый код)

Работа с Точками Останова в C/AL Editor

Чтобы переключаться между настройками: включить, отключить и удалить точку останова из C/AL Editor можно использовать клавишу F9 (или Сервис, Debugger, Установить Точку Остановки). Информация о точках останова сохраняется в виртуальной таблице ***Breakpoint*** при закрытии C/AL Editor.

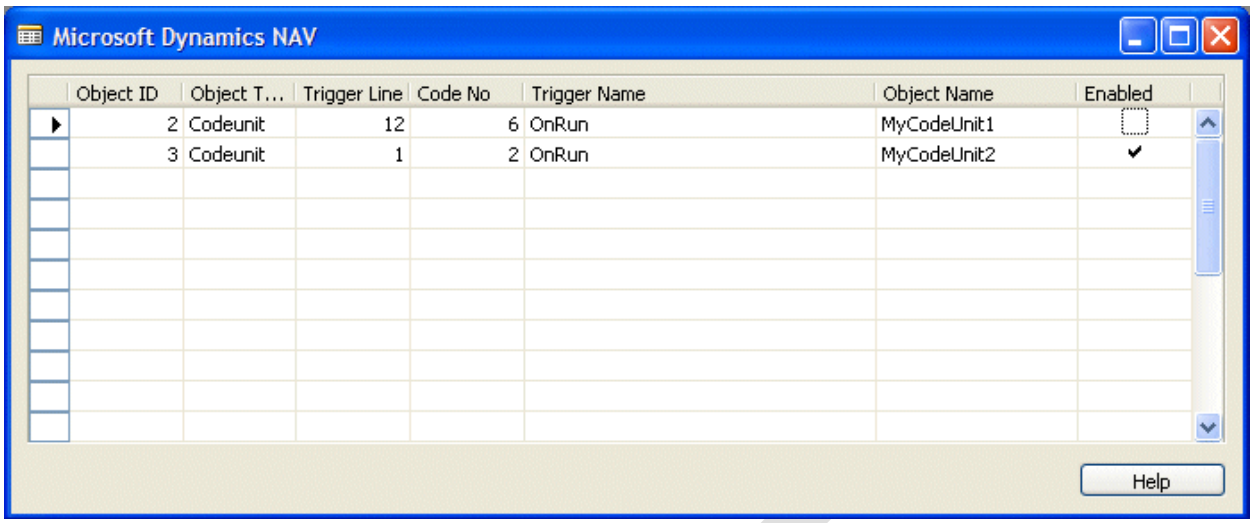
Виртуальная Таблица Breakpoint

Виртуальная таблица ***Breakpoint*** (ID 2000000059) может хранить следующую информацию об установленных точках останова:

Поле	Описание
Object ID	Идентификатор объекта, для которого хранится информация о точках останова
Object Type	Table, Form, Report, Dataport или Codeunit.
Trigger Line	Номер строки триггера, для которой установлена точка останова.
Code No.	Номер триггера, который содержит точки останова. Среда C/SIDE использует этот номер для идентификации триггера в момент выполнения
Trigger Name	Название триггера, в котором установлена точка останова
Object Name	Названием объекта
Enabled	В данном поле устанавливается Флаг для включенных точек останова.

Чтобы управлять точками останова в виртуальной таблице ***Breakpoint,*** нужно создать форму. Ниже приведен пример того, как эта форма должна выглядеть:

DRAFT



Информация о точках останова сохраняется при закрытии объекта или при сохранении нового объекта. Компиляция не нужна. Точки останова для не сохраненных объектов не хранятся.

Хранение Точек Останова в XML Файле

Точки останова, хранящиеся в виртуальной таблице ***Breakpoint,*** автоматически сохраняются в файл NaviBP.xml. По умолчанию файл хранится в той же папке, что и файл fin.zup. Для компьютеров под управлением Windows XP путь будет следующим:

C:\Documents and Settings\user\Application Data.

Ниже представлен пример XML файла, содержащего информацию о точках останова для двух объектов:

```
  <?xml version="1.0" ?>
- <BreakpointList>
  + <Object Type="Codeunit" ID="2" Name="MyCodeunit1">
  + <Object Type="Codeunit" ID="3" Name="MyCodeunit2">
  </BreakpointList>
```

Этот файл содержит точки останова для MyCodeunit1 и MyCodeunit2. Объекты показываются как элемент XML с названием "Object". У элемента object три атрибута: Type, ID и Name.

Если развернуть первый объект, MyCodeunit1, то можно увидеть один элемент "Breakpoint". Это означает, что для объекта установлена одна точка останова:

```
  <?xml version="1.0" ?>
- <BreakpointList>
  - <Object Type="Codeunit" ID="2" Name="MyCodeunit1">
    - <Breakpoint>
        <TriggerName>OnRun</TriggerName>
        <CodeNo>6</CodeNo>
        <TriggerLine>12</TriggerLine>
        <Enabled>No</Enabled>
      </Breakpoint>
    </Object>
  + <Object Type="Codeunit" ID="3" Name="MyCodeunit2">
  </BreakpointList>
```

Если полностью развернуть элемент breakpoint, то для каждой точки останова можно увидеть следующую информацию:

Тэг XML	Описание

Тэг XML	Описание
TriggerName	Название триггера, в котором установлена точка останова
CodeNo	Номер триггера, который содержит точки останова. Среда C/SIDE использует этот номер для идентификации триггера в момент выполнения.
Trigger Line	Номер строки триггера, для которой установлена точка останова.
Enabled	Выражение типа Boolean показывает, включена ли точка останова

Запуск Dynamics NAV или NAV Application Server с Использование Другого Файла с Точками Останова

Как Dynamics NAV, так и NAV Application Server можно запустить с параметром breakpoints. Это позволит указать определенный файл для сохранения и загрузки информации о точках останова.

Пример

FIN.EXE breakpoints=C:\MyBreakpoints.xml

Хранение Отладочной Информации в Файле FIN.ZUP

Настройки, сделанные с помощью команд: Сервис, Debugger, Active и Сервис, Debugger, Точка Останова на Триггерах хранятся в файле fin.zup. Это означает, что если отладчик был активен и срабатывал на триггерах, когда был выполнен выход, то эти же настройки будут применяться при следующем входе в систему.

Обзор Горячих Клавиш

Ниже приведен список горячих клавиш для часто используемых при отладке команд:

Горячая Клавиша	Команда
SHIFT+CTRL+F11	Debugger Active
F5	Go
F9	Toggle Breakpoint
SHIFT+CTRL+F12	Breakpoint on Triggers
SHIFT+F9	Open Breakpoints Dialog Box
CTRL+SHIFT+F9	Remove All Breakpoints
F8	Step Into
CTRL+F8	Step Over
ALT+NUM*	Show Next Statement
SHIFT+F5	Stop Debugging

Отладчик и Буфер Команд

C/SIDE использует буфер команд для улучшения производительности. Но при запуске отладчика C/SIDE отключает буфер команд. Дополнительную информацию см. в главе ГЛАВА 30 ПРОИЗВОДИТЕЛЬНОСТЬ.

DRAFT

18.3 Инструмент Код Приложения

Если в программный модуль 1 (Codeunit 1) добавить функцию (триггер) с идентификатором 6, то вы сможете получить доступ к функциональности Код Приложения из меню Сервис, Debugger. Отсюда можно запускать и останавливать протоколирование выполнения кода. Также можно просматривать запротоколированный код. Кроме того, можно использовать функцию CODECOVERAGELOG, чтобы запускать и останавливать протоколирование кода непосредственно из кода. Также данная функция возвращает текущий статус протоколирования. Дополнительную информацию о данной функции см. в онлайновой справке *C/SIDE Reference Guide*.

Функциональность Код Приложения полезна при выполнении модификаций Dynamics NAV, когда нужно проверить выполненные доработки. Она предоставляет быстрый обзор объектов, чей код был выполнен, также она может отображать этот код.

Окно ***Код Приложения*** отображает объекты (таблицы, формы, отчеты, датапорты и программные модули), чей код был выполнен при выполнении одной или нескольких транзакций. Окно ***Код Приложения Обзор*** отображает для выбранных объектов, запротоколированный код. В следующем разделе будет приведено описание окон ***Код Приложения*** и ***Код Приложения Обзор***.

Использование Инструмента Код Приложения

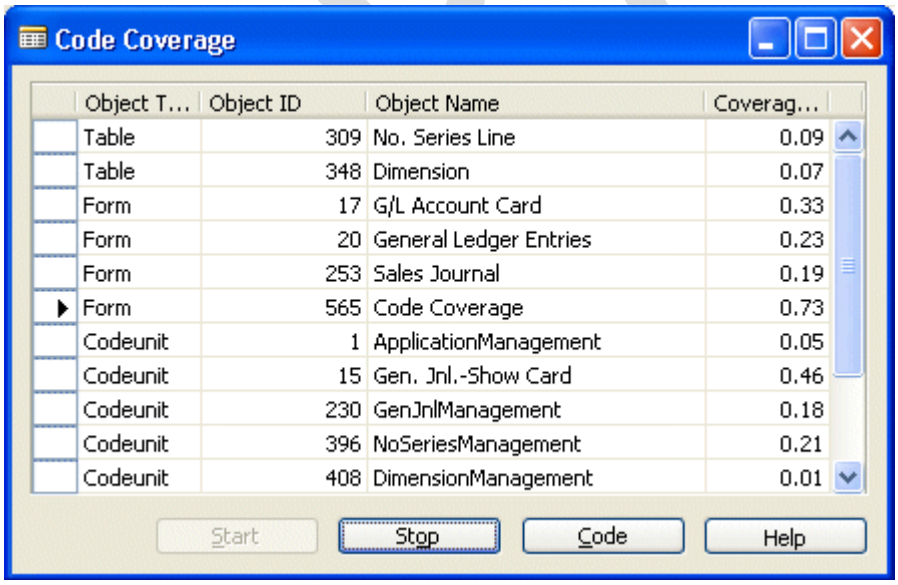
Как было сказано ранее, инструмент Код Приложения полезен для обзора объектов, которые выполнялись, при выполнении пользователем каких-либо операций, а также для просмотра кода, который использовался в данных транзакциях.

Чтобы осуществить протоколирование выполняемого кода:

1. Щелкните Сервис, Debugger, Выполнение Кода. Появится окно ***Код Приложения***.

2. Щелкните кнопку Старт. Теперь инструмент Код Приложения готов к протоколированию кода.

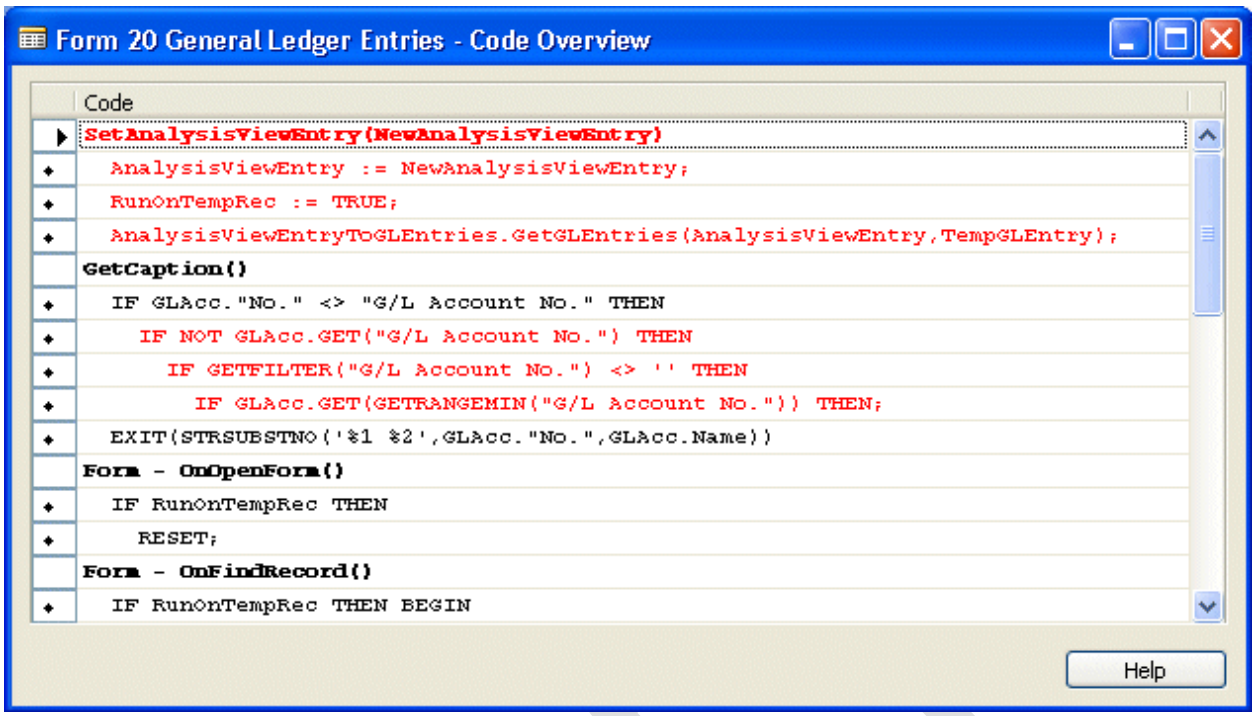
3. Теперь нужно выполнить операции, которые нужно исследовать, затем вернуться в окно ***Код Приложения***. Теперь в нем содержится список всех таблиц, форм, отчетов, датапортов и программных модулей, которые были использованы при выполнении операций.



Object T...	Object ID	Object Name	Coverag...
Table	309	No. Series Line	0.09
Table	348	Dimension	0.07
Form	17	G/L Account Card	0.33
Form	20	General Ledger Entries	0.23
Form	253	Sales Journal	0.19
Form	565	Code Coverage	0.73
Codeunit	1	ApplicationManagement	0.05
Codeunit	15	Gen. Jnl.-Show Card	0.46
Codeunit	230	GenJnlManagement	0.18
Codeunit	396	NoSeriesManagement	0.21
Codeunit	408	DimensionManagement	0.01

4. Щелкните кнопку Стоп.

5. Выберите объект, чей код нужно просмотреть. Щелкните кнопку **Код**, чтобы открыть окно ***Код Приложения Обзор***:



В окне ***Код Приложения Обзор*** отображается код объекта, выбранного в окне ***Код Приложения***. Строки кода, выполненные при выполнении транзакции, отображаются черным. Строки кода, которые не выполнялись, показаны красным.

Окно ***Код Приложения Обзор*** отображает код таким же образом, как и отладчик. Но в отладчике код показывается во время выполнения, а в окне ***Код Приложения*** показывается конечный результат (выполненный код). Слева от строки с кодом, который можно выполнить отображается специальный маркер. Корректный цвет имеют только строки с маркером, строки в которых нет маркера, отображаются цветом соседней строки.

Важно

При использовании инструмента Выполнение Кода нельзя выполнять изменение объектов, это может привести к непредсказуемым результатам.

DRAFT

ГЛАВА 19 РАСШИРЕНИЕ ВОЗМОЖНОСТЕЙ ЯЗЫКА C/AL

В данной главе будет описано, как расширить возможности языка C/AL с помощью технологий COM.

- Что такое COM?
- Использование технологий COM в среде C/SIDE
- Использование C/SIDE как Automation Controller
- Получение событий в C/SIDE
- Использование компонентов (Custom Controls) из среды C/SIDE

19.1 Что Такое COM?

В настоящем руководстве не будет приведено описание технологий COM, OCX, Automation, OLE, ActiveX и т.п.. Данная тема крайне большая и сложная, кроме того по ней существует ряд хороших книг.

COM и C/SIDE

В C/SIDE можно использовать технологии COM двумя способами: можно использовать OCX и Automation (C/SIDE в роли automation controller или клиента). Существует множество коммерческих OCX, с помощью которых можно решать различные задачи, также можно разработать свои собственные компоненты OCX. Если среда C/SIDE используется в роли automation controller, это позволяет работать, например, с программами из поставки Microsoft Office.

Если вы намереваетесь создавать свои компоненты, вероятно для этого вы используете Microsoft Visual C++ или Microsoft Visual Basic. В обоих продуктах используются мастера, которые упрощают разработку COM объектов. Это позволяет выполнять разработку COM, не разбираясь в тонкостях технологии COM. Если вы собираетесь использовать существующие COM объекты (controls или automation servers) из среды C/SIDE, вам нет необходимости, знать что-либо о технологии COM. Использование функциональности, предоставляемой COM объектами, не отличается от использования функций C/AL.

Ниже приведен список рекомендуемой литература для тех, кто хочет знать больше:

- David Chappell. Understanding ActiveX and OLE. Microsoft Press (1996). (Общий обзор, не вдаваясь слишком глубоко в детали)
- Dale Rogerson. Inside COM. Microsoft Press (1997). (Книга содержит много технических описаний)
- Kraig Brockschmidt. Inside OLE, 2nd edition. Microsoft Press (1995). (Очень подробная книга, предназначена для тех, кто хочет знать все тонкости, но при этом самая старая из упомянутых книг)

19.2 Использование Технологий COM в Среде C/SIDE

В C/SIDE можно использовать технологии COM двумя способами: можно использовать OCX и Automation (C/SIDE в роли automation controller или клиента). При этом существует ряд ограничений:

Поддерживаются только не визуальные компоненты. Это означает, что в среду C/SIDE нельзя добавить графические объекты (например, нельзя добавить новый элемент управления на форму). Однако компоненты могут отображать информацию и взаимодействовать с пользователями с помощью своих собственных окон.

Обработка исключений. C/SIDE не поддерживает получение информации о возникших исключениях в компонентах или automation server с помощью методов Invoke интерфейса IDispatch

и структуры EXCEPINFO (как описано в книге *Inside OLE)*. Как обойти данное ограничение будет описано далее.

Параметры, Возвращаемые Значения и Типы Данных

Как можно увидеть в литературе про COM, механизм вызова методов компонентов, передача параметров и получение возвращаемых значений достаточно сложный. Использование мастеров в Microsoft Visual C++ ограничит вас от большинства трудностей.

Однако вам следует знать, что между типами данных, используемых в методах Visual C++ и типами данных C/AL нет точного соответствия. Некоторые типы данных COM не поддерживаются в C/AL, а для некоторых существуют ограничения в использовании.

При использовании C/AL Symbol Menu можно увидеть синтаксис метода или свойства, а также возвращаемое значение и параметры, с указанием типов данных COM.

В следующей таблице приведены соответствующие типы данных COM для типов данных C/AL:

Тип данных C/AL	Тип данных COM	Примечание
Boolean	VARIANT_BOOL (VT_BOOL)	
Option	long (VT_I4)	
Integer	long (VT_I4)	
BigInteger	long (VT_14)	
Decimal	CURRENCY (VT_CY)	Тип данных CURRENCY в COM это специальный тип данных с фиксированной точкой, у которого 15 разрядов слева от точки и четыре справа. Для типа Decimal в C/AL не используется фиксированная точка, а всего в нем 18 разрядов. Это может привести к выполнению округления. Сервер будет обрабатывать переданное значение как CURRENCY, и вернет его также как CURRENCY. Не имеет значения, сколько знаков в оригинальном значении Decimal было справа от точки, у возвращенного значения типа CURRENCY будет не больше четырех разрядов.
Char	BSTR (VT_BSTR)	
Text	BSTR (VT_BSTR)	
Code	BSTR (VT_BSTR)	
Date	DATE (VT_DATE)	
Time	void (VT_VOID)	
DateTime	DATE (VT_DATE)	
Automation	TypedObject, UntypedObject (VT_DISPATCH)	
InStream	VT_STREAM	
OutStream	VT_STREAM	
Variant	VARIANT (VT_VARIANT)	

В следующей таблице приведены соответствующие типы данных C/AL для типов данных COM:

Тип Данных COM	Тип Данных C/AL	Примечание

DRAFT

Тип Данных COM	Тип Данных C/AL	Примечание
VT_UNKNOWN	InStream или OutStream	Поддерживаются только интерфейсы IID_IStream и IID_SequentialStream. Если передать другой интерфейс, например, IUnknown, то возникнет ошибка выполнения.
short (VT_I2)	Integer	
long (VT_I4)	Integer, BigInteger	
float (VT_R4)	Decimal	
double (VT_R8)	Decimal	
CURRENCY (VT_CY)	Decimal	Тип данных CURRENCY в COM это специальный тип данных с фиксированной точкой, у которого 15 разрядов слева от точки и четыре справа. Для типа Decimal в C/AL не используется фиксированная точка, а всего в нем 18 разрядов
DATE (VT_DATE)	Date DateTime	Тип данных COM DATE одновременно содержит и дату и время. В C/AL Date и Time это разные типы данных. Поэтому при использовании типа данных Date в C/AL информация о времени будет потеряна. Чтобы получить и дату, и время нужно использовать в C/AL тип данных DateTime.
BSTR (VT_BSTR)	Text	
VARIANT_BOOL (VT_BOOL)	Boolean	
TypedObject/ UntypedObject (VT_DISPATCH)	Automation/OCX	
VT_EMPTY	Text	
VARIANT (VT_VARIANT)	Variant	

Unsigned char (VT_UI1), SCODE (VT_ERROR) и SAFEARRAY (VT_ARRAY)

Для передачи unsigned char, SCODE и SAFEARRAY можно использовать тип данных C/AL variant. Эти значения нельзя присвоить типам данных C/AL, только передать другим variant.

Дополнения

При вызове метода с параметром ByRef, обычное преобразование типов языке C/AL не работает. Это значит, что если у параметра тип данных float, то нужно использовать переменную C/AL с типом данных Decimal. Нельзя использовать Integer в надежде, что C/AL выполнить преобразование. Совет: в данном случае присвойте значение переменной типа integer переменной с типом decimal, а после это передайте в качестве параметра переменную с типом Decimal.

Иногда в C/AL Symbol Menu будет показано, что метод или свойство COM объекта используют тип IDispatch. Это означает, что метод или свойство возвращают или получают в качестве параметра COM объект. В этом случае нужно использовать переменную с типом Automation, для которой указан корректный COM объект (поле Subtype). Чтобы получить необходимую информацию, следует обратиться к документации для данного automation server.

Также можно увидеть методы и свойства, которые не относятся к «обычным» типам данных.

DRAFT

Например, метод Microsoft Excel может вернуть значение типа WORKBOOK.

Это означает, что automation server использует тип USERDEF. Среда C/SIDE поддерживает тип USERDEF в двух контекстах: IDispatch и Enumeration.

Если тип USERDEF является IDispatch, в этом случае он является интерфейсом (также называется, классом или объектом) с определенным GUID. В этом случае нужно использовать такой же тип объекта в качестве параметра или возвращаемого значения. Для этого нужно создать переменную с типом Automation и корректным Subtype.

Например, ряд методов Microsoft Excel возвращает переменную WORKBOOK. Это значит, что нужно объявить переменную с типом Automation и subtype 'Microsoft Excel 8.0 Object Library'.Workbook.

Если тип USERDEF является Enumeration, то следует знать, что вы не можете использовать символьные имена (например, xl3DPie), а вместо этого должны использовать enumerator (например, -4102). Для продуктов Microsoft Office эти значения можно найти в VBA Object Browser.

19.3 Использование C/SIDE как Automation Controller

Ниже приведено общее описание процедуры использования automation server из среды C/SIDE. Для использования automation server нужно выполнять пять шагов:

1. Объявить создаваемый (top-level) интерфейс (class) automation server как переменную типа Automation.

2. Объявить остальные интерфейсы (классы) как переменные типа Automation.

3. Использовать C/AL функцию CREATE для переменной, объявленной на шаге 1. Не использовать CREATE для других переменных.

4. Использовать методы и свойства automation server в C/AL.

5. При необходимости можно CLEAR (destroy) top-level объект. Он автоматически будет уничтожен, когда переменная покинет область видимости.

Большая часть кода будет написана на шаге 4, в ней будут использованы методы и свойства automation server. Синтаксис данных методов и свойств описан в документации к automation server. Использование этих методов и свойств не требует нового или измененного синтаксиса в C/AL коде. В двух следующих разделах будет показано, как использовать Microsoft Word и Microsoft Excel.

Создание Письма в Microsoft Word

В данном примере будет внедрена функциональность создания письма в Microsoft Word. Данная функциональность будет инициироваться щелчком по кнопке в карточке клиента (Customer card). Письмо будет создано только в том случае, если клиент приобрел товаров за прошлый год больше чем на 2.500 LCY (локальная валюта). В этом случае клиенту будет предложено воспользоваться 3% скидкой.

Большая часть информации, которую нужно передать в Word находится в таблице ***Customer***. Сюда относится информация, используемая для заголовка письма, такая как название и адрес клиента, а также имя контактного лица, которому адресуется письмо.

В таблице ***Customer*** есть вычисляемое поле (FlowField) **Sales (LCY)**, которое содержит требуемую финансовую информацию – сводные продажи клиенту. Для данного примера (цель которого – демонстрация использования automation) данное значение будет использоваться, как оно есть. Обратите внимание, что это не то, как надо делать в реальной жизни. Для правильного расчета суммы продаж за прошлый год, нужно установить корректный фильтр (date filter).

Также потребуется получить информацию о вашей компании, чтобы использовать ее в заголовке. Данная информация содержится в таблицах ***Company Information*** и ***User***.

DRAFT

Куда Поместить Automation Код

Весь код будет помещен в отдельный программный модуль, который будет вызываться при выборе элемента меню в карточке клиента (Customer card).

При принятии решения о том, куда следует поместить код с использованием automation, следует принять во внимание два момента:

- Во-первых, automation server должен быть установлен на компьютере, с которого будет выполнять компиляция объекта, содержащего этот automation. Если вам потребуется изменить и перекомпилировать объект на компьютере, на котором не установлен automation server, то чтобы иметь возможность скомпилировать объект придется внести в код значительные изменения. По этой причине, рекомендуется код, содержащий объект automation, помещать в отдельные программные модули.
- Вторым моментом является производительность. При создании automation server (CREATE) наблюдаются значительное снижение производительности. Поэтому если automation server будет использоваться несколько раз, то чтобы улучшить производительность нужно проектировать код так, чтобы создание automation server выполнялось только один раз (вместо серии вызовов функций CREATE/CLEAR).

К сожалению, эти два момента противоречат друг другу, поэтому при принятии решения о том, куда помещать код придется идти на компромисс, согласно условиям, в которых он будет использоваться.

В данном примере решено не помещать код, содержащий automation, в карточку клиента, а поместить его в отдельный программный модуль. Производительность можно будет улучшить, поместив в карточке клиента (Customer card) такой код, чтобы не требовалось открывать и закрывать Word для каждого письма в одной сессии.

Однако для данного метода есть своя проблема: если Word уже открыт, то при вызове его из кода, работающий экземпляр будет использован повторно. Это значит, что пользователь может открыть Word "вручную" или не закрыть после того, как будет создано первое письмо.

Создание Шаблона в Word

Первым делом нужно подготовить шаблон, который будет использоваться при создании писем клиентам.

Чтобы создать шаблон:

1. Откройте Word и откройте новый документ.

2. Щелкните Insert, Field и добавьте поле MergeField из категории Mail Merge.

3. Введите *Contact* в качестве названия поля.

Простой шаблон, который вы собираетесь использовать в данном примере должен содержать следующие поля:

Название Поля	Содержит	Соответствующая таблица
Name	Название клиента	Customer
Address	Адрес клиента	Customer
Sales (LCY)	Сумма покупок клиента	Customer
Contact	Имя контактного лица со стороны клиента	Customer
Company Name	Название вашей компании	Company Information
User Name	Имя лица, подписывающего письмо.	User

DRAFT

4. Создайте указанные шесть полей и сохраните документ Word как Discount.dot.

Создание Программного Модуля и Объявление Переменных

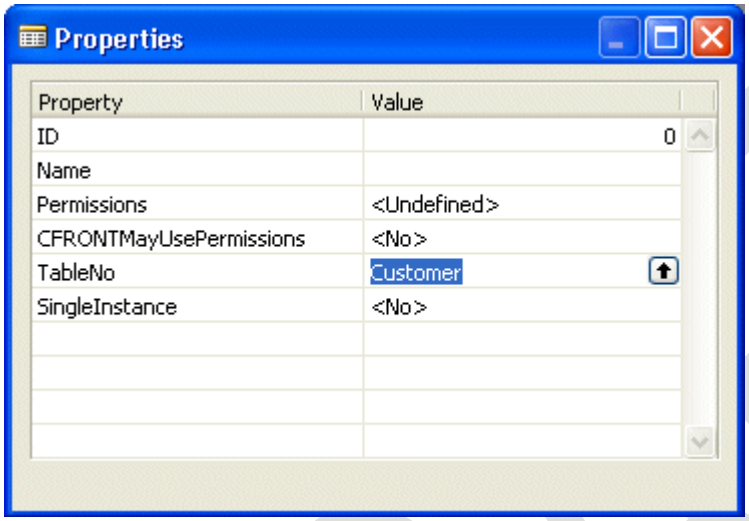
Следующим шагом будет создание программного модуля, в котором будет происходить вызов Word и создание письма.

1. Откройте **Object Designer** (SHIFT+F12), щелкните кнопку **Codeunit**, затем кнопку **New**, чтобы создать новый программный модуль.

2. Щелкните Вид, Properties (SHIFT+ F4), чтобы открыть окно ***Properties*** для программного модуля.

3. Щелкните кнопку AssistButton в свойстве *TableNo*, чтобы открыть окно ***Table List***.

4. В окне ***Table List*** выберите ***Customer*** и щелкните кнопку OK:



Установив в свойстве *TableNo* значение ***Customer***, вы установили очень тесную связь между карточкой клиента (Customer card) и данным программным модуле. Позже в карточку клиента будет добавлен специальный элемент меню, из которого будет вызываться данный программный модуль. При вызове программного модуля в него будет передаваться значение текущей записи. Для этого не потребуется предпринимать никаких специальных действий.

Объявление переменных

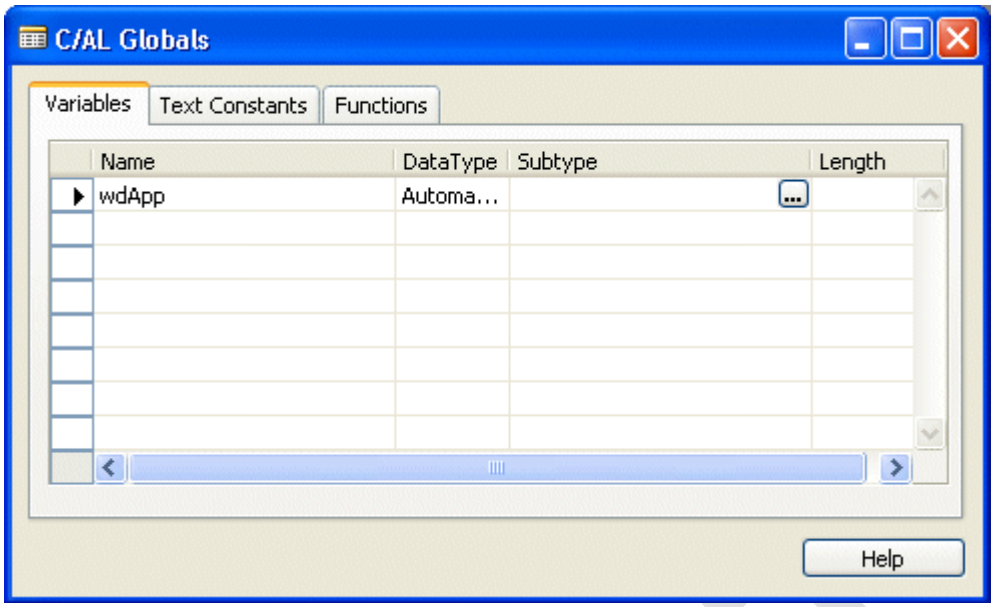
Теперь нужно объявить требуемые переменные.

1. Щелкните Вид, C/AL Globals.

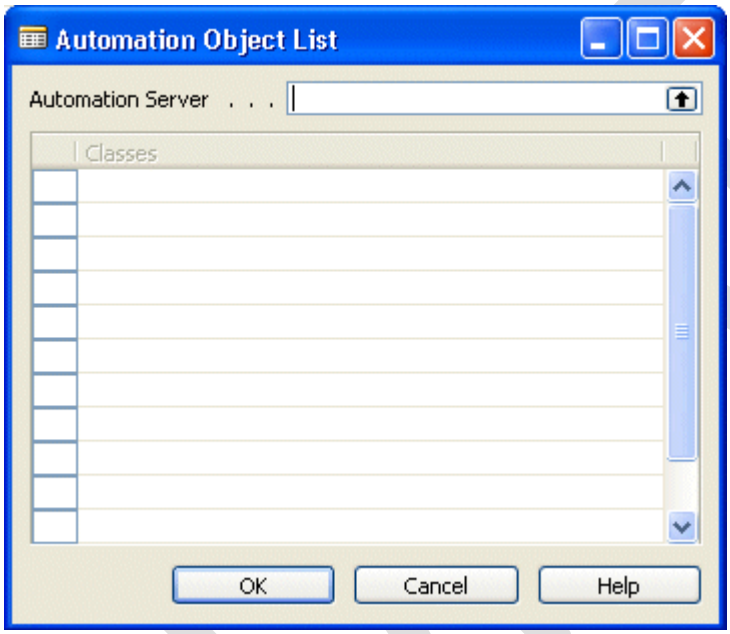
2. Первой будет объявлена переменная для top-level (создаваемого) класса Word. Название класса - Application.

Введите название переменной *wdApp,* выберите тип данных Automation.

DRAFT

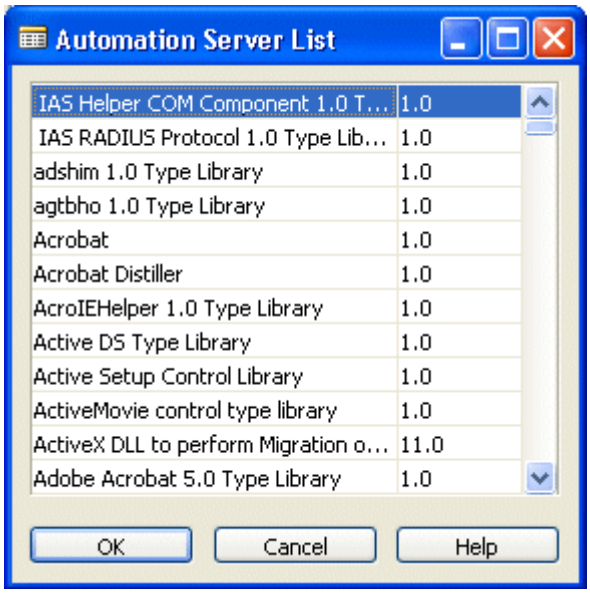


3. В поле ***Subtype*** щелкните кнопку AssistButton, чтобы открыть ***Automation Object List***:



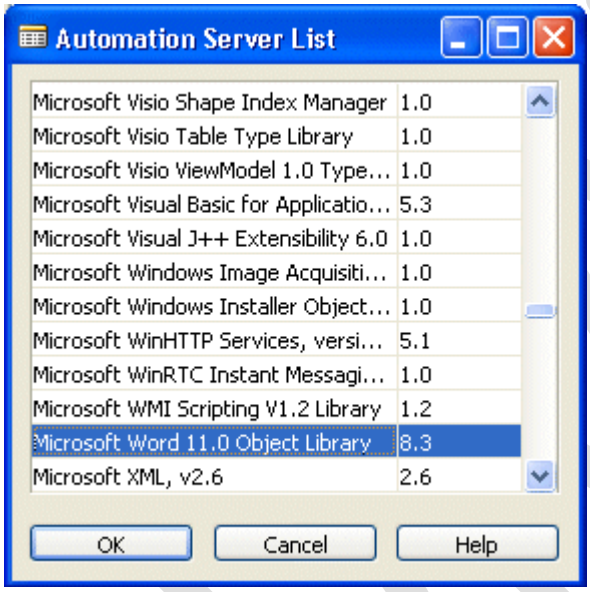
Это окно используется для выбора классов automation server. Но вначале нужно выбрать сам automation server.

4 Щелкните кнопку AssistButton в поле **Automation Server**, чтобы открыть окно ***Automation Server List***:



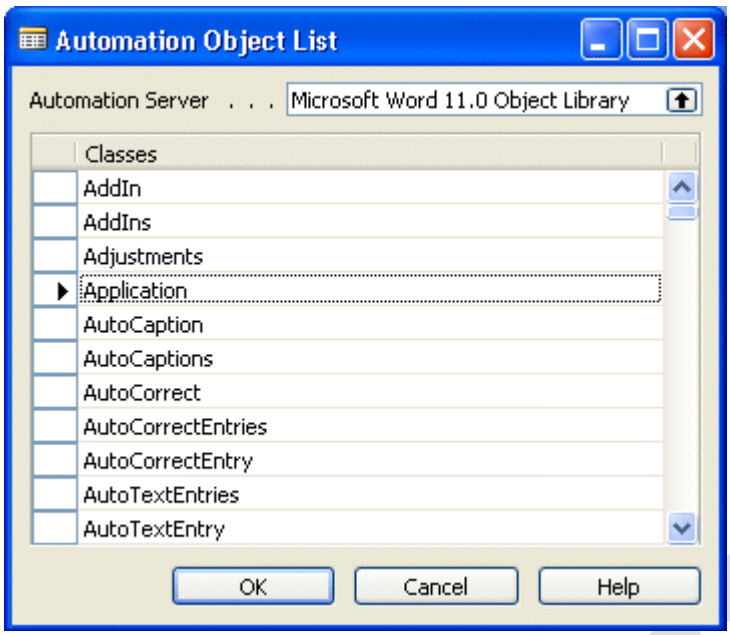
Это список всех automation server установленных на данном компьютере.

Прокрутите вниз к Microsoft Word и выберите его:



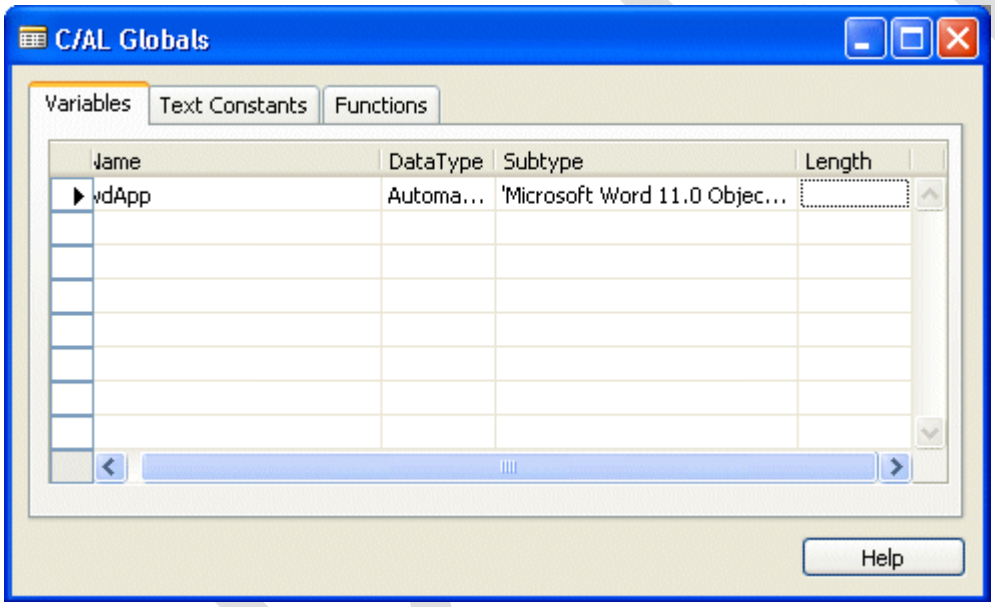
5. Щелкните кнопку OK чтобы закрыть окно ***Automation Server List***. Теперь в окне ***Automation Object List*** доступен список всех классов Microsoft Word 11.0 Object Library:

DRAFT



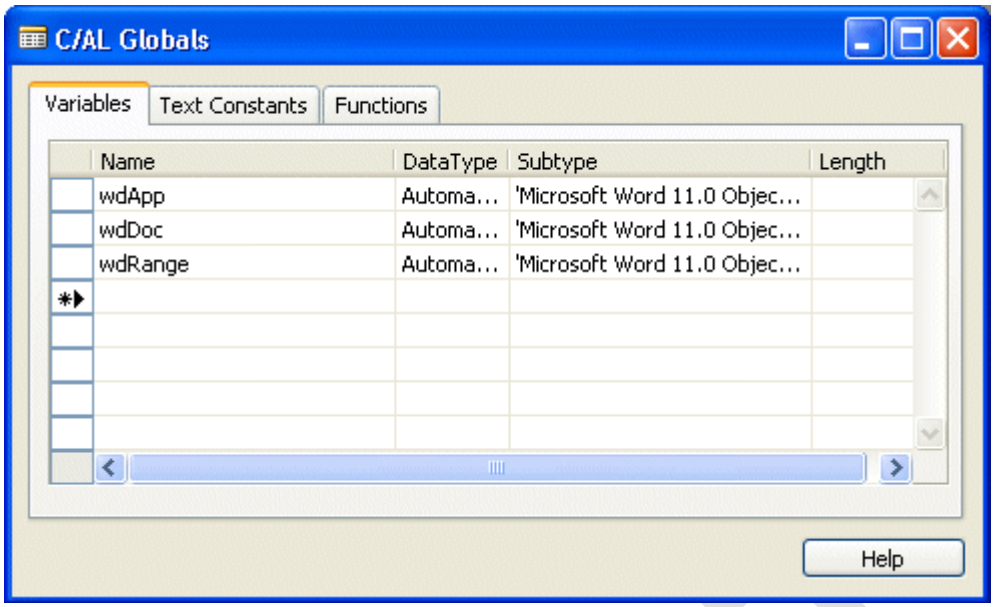
6. Выберите Application и щелкните кнопку OK.

Теперь определена переменная с создаваемым (top-level) классом Word:



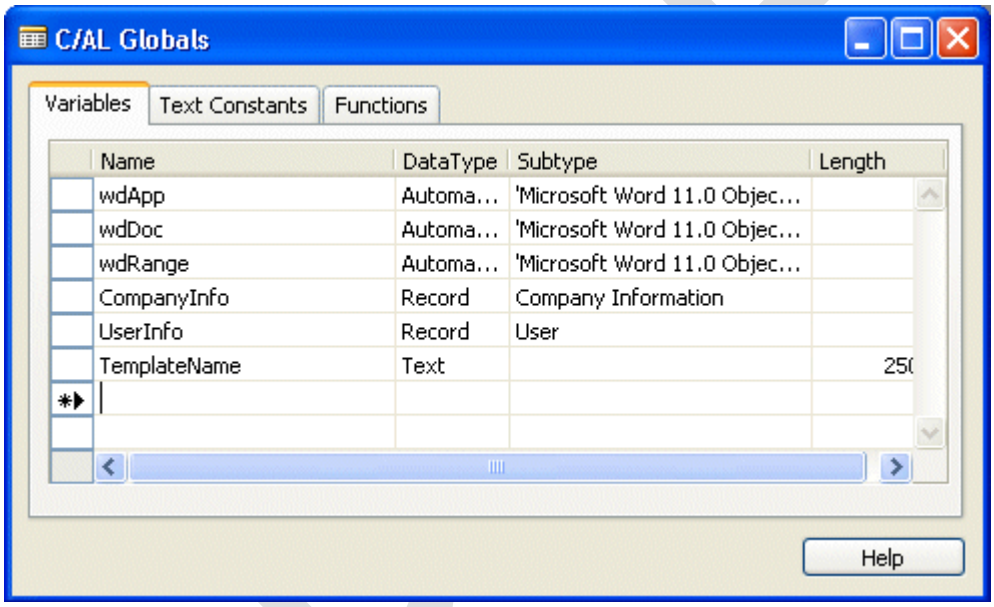
7. Для данного примера нужно еще два класса Word: Document и Range. Объявите переменные таким же образом как с переменной Application.

Окно ***C/AL Globals*** в результате должно выглядеть следующим образом:



8. Также потребуется объявить еще несколько переменных: две переменных типа record, которые указывают на таблицы ***Company Information*** и ***User***, а также текстовую переменную, для хранения названия шаблона Word на основании, которого будут создаваться письма.

Окно ***C/AL Globals*** должно выглядеть следующим образом:



Изменить длину текстовой переменной TemplateName с 30 до 250.

Написание C/AL Кода

Перед тем как начать писать код с использованием automation нужно выполнить некоторые подготовительные действия.

Подготовительные действия

Следует начать с расчеты значения вычисляемого поля **Sales (LCY)**. Затем нужно выяснить должна ли клиенту предоставляться скидка. Далее получить информацию из таблиц ***Company Info*** и ***User***, которая будет использоваться для заполнения некоторых полей в письме.

```
CALCFIELDS("Sales (LCY)");

IF ("Sales (LCY)" < 2500) THEN

  EXIT;
```

CompanyInfo.FIND;

UserInfo.GET(USERID);

Создание automation server

Перед тем как использовать экземпляр Word его нужно создать. Для этого используется функция CREATE. Она вызывается следующим образом:

CREATE(wdApp)

Для функции CREATE можно передавать необязательный параметр NewServer, у которого значение по умолчанию FALSE. Это означает, что если экземпляр automation server уже запущен, то он будет использован. Если установить значение параметра NewServer в TRUE, как в CREATE(wdApp, TRUE), то будет создан новый экземпляр Word. Если для вашего приложения важно можно ли управлять способом создания automation server (указывать создавать новый экземпляр или использовать существующий), то обратитесь к документации automation server.

Добавление нового документа

Теперь добавим новый документ в Word, данный документ будет использовать шаблон созданный ранее:

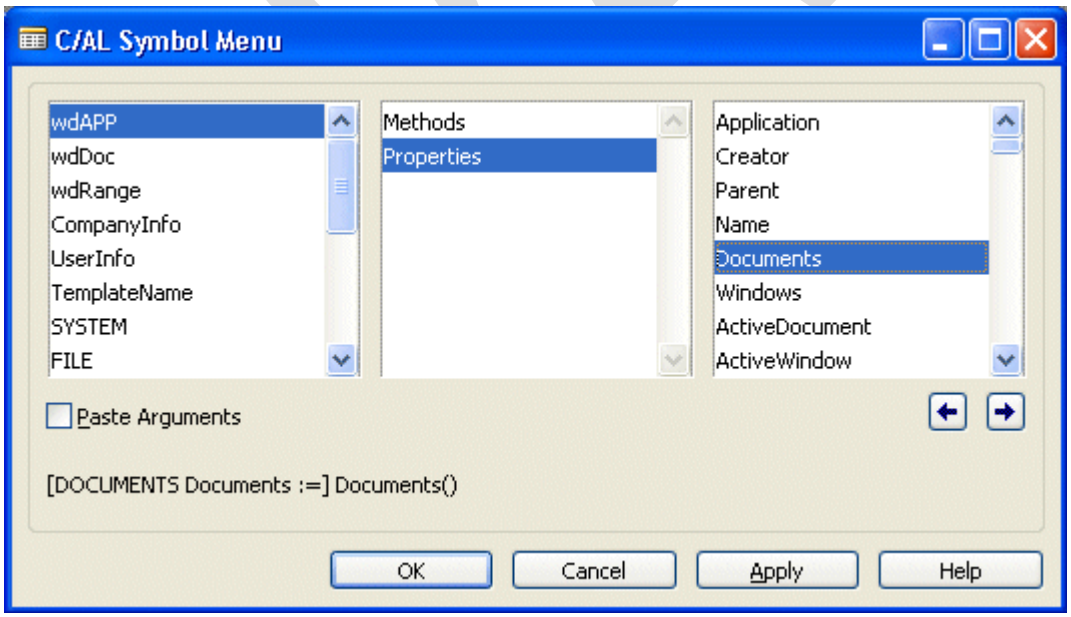
TemplateName := 'C:\Documents and Settings\All Users\Discount.dot

wdDoc := wdApp.Documents.Add(TemplateName)

wdApp.ActiveDocument.Fields.Update

Так как метод Add из Documents collection требует, чтобы путь к шаблону был передан в него по ссылке, нужно присвоить переменной TemplateName данную информацию. Если попытаться передать параметр простой строкой, но на этапе компиляции возникнет ошибка.

Посмотрим на синтаксис свойства Documents переменной wdApp (класс Microsoft Word Application):



Если посмотреть в онлайновой справке по Microsoft Word Visual Basic описание свойства Documents, то можно увидеть, что свойство Documents возвращает Documents collection, содержащую все открытые документы. Так же можно посмотреть, что у объекта Documents collection есть метод Add, и синтаксис метода Add следующий:

expression.Add(Template, NewTemplate, Document Type, Visible)

где *expression* обязательный параметр, который возвращает значение объект Documents. Все остальные параметры являются необязательными. Чтобы открыть новый документ на основании шаблона, мы используем свойство Template.

Еще раз посмотрим на синтаксис в **C/AL Symbol Menu**. Обратите внимание, что свойство Documents возвращает объект типа DOCUMENTS. Это означает, что свойство возвращает класс Documents (интерфейс IDispatch).

```
wdDoc := wdApp.Documents.Add(TemplateName)
```

Так как метод Add возвращает класс Document, то для него была объявлена переменная C/AL - wdDoc.

Третья строка (wdApp.ActiveDocument.Fields.Update;) содержит вызов, который должен гарантировать, что шаблон работает как следует.

Передача данных в Word

Теперь вы готовы передать фактические данные из записи Customer в поля документа Word.

Теперь нужно настроить три первых поля в шаблоне, чтобы они стали содержать поля contact, name и address клиента, для этого нужно перемести данные с помощью следующего кода:

```
wdRange := wdAPP.ActiveDocument.Fields.Item(1).Result;
wdRange.Text := Contact;
wdRange.Bold := 1;
wdRange := wdAPP.ActiveDocument.Fields.Item(2).Result;
wdRange.Text := Name;
wdRange.Bold := 1;
wdRange := wdAPP.ActiveDocument.Fields.Item(3).Result;
wdRange.Text := Address;
wdRange.Bold := 1;
```

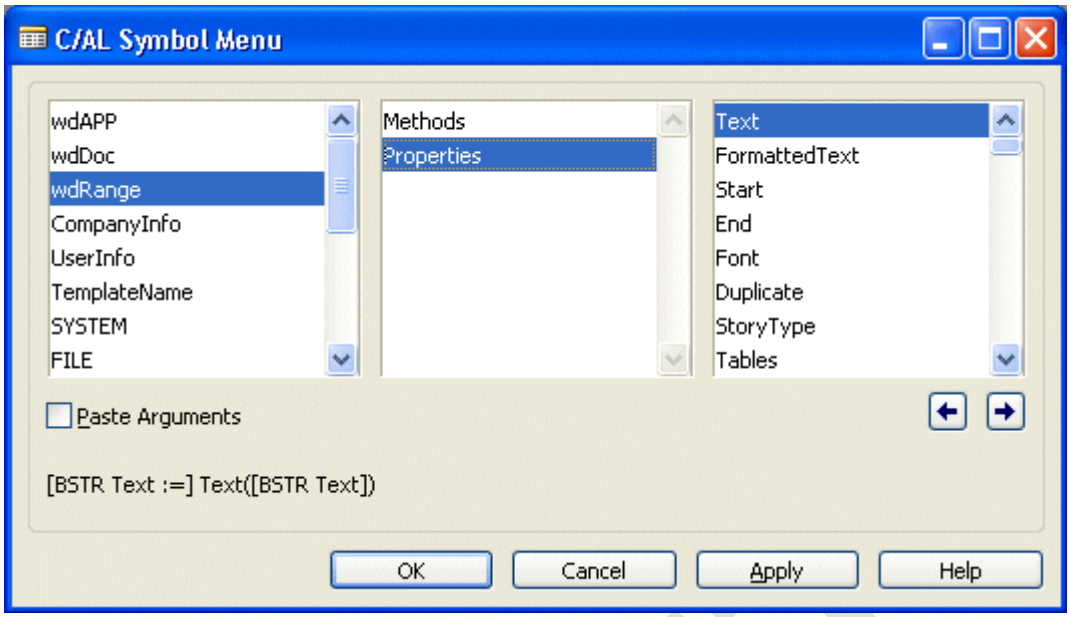
Вы не можете присваивать значения переменных, непосредственно используя присваивание типа...Fields.Item(3) := Address. Вместо этого нужно использовать свойство *Result* поля. Это свойство возвращает result поля как range. Этот range нужно поместить в третью объявленную переменную типа automation - wdRange.

Затем можно установить в свойство *Text* переменной wdRange требуемое значение – имя контактного лица, название и адрес клиента. В конце будет включено форматирование жирным, чтобы выделить текст.

> **Использование Методов По Умолчанию**
>
> Документация к Microsoft Word Visual Basic использует следующий синтаксис: wdApp.ActiveDocument.Fields(3).Result
>
> Как видно, он несколько отличается от синтаксиса, используемого в примере: wdApp.ActiveDocument.Fields.Item(3).Result
>
> Причина заключается в том, что метод Item является методом по умолчанию для Fields collection. Visual Basic использует этот метод, если программист не укажет альтернативный метод. Однако в среде C/SIDE нужно указывать точное название метода Item.

Нужно помнить, что передаваемые данные должны быть в текстовом формате. Если данные другого формата, то будет получена ошибка компиляции. Как видно на следующем рисунке wdRange.Text ожидает аргумент с типом данных BSTR, который соответствует типам данных C/SIDE Text или Code:

DRAFT



Это означает, что все данные типов данных отличных от Text или Code перед отправкой в Word должны быть преобразованы.

В данном примере нужно передать в Word значение поля **Sales (LCY).** Это поле типа Decimal, поэтому для преобразования его в Text нужно использовать функцию FORMAT. Синтаксис функции FORMAT:

String: = FORMAT (Value [, Length] [, FormatNumber | FormatString])

Преобразование и передача в Word данных из поля **Sales (LCY)** выполняется следующим кодом:

```
wdRange := wdAPP.ActiveDocument.Fields.Item(4).Result;
wdRange.Text := FORMAT("Sales (LCY)");
wdRange.Bold := 0;
```

Теперь остается передать данные из остальных таблиц. Следующий код выполняет передачу данных полученных из таблиц ***Company Information*** и ***User***:

```
wdRange := wdApp.ActiveDocument.Fields.Item(5).Result;
wdRange.Text := CompanyInfo.Name;
wdRange := wdApp.ActiveDocument.Fields.Item(6).Result;
wdRange.Text := UserInfo.Name;
```

Завершение кода

После того как все данные были переданы в Word, нужно ввести еще две строки кода для завершения обработки:

```
wdApp.Visible := TRUE;
wdApp.ActiveDocument.Fields.Unlink;
```

В первой строке выполняется открытие Word и отображения созданного письма. Вторая строка преобразует значения полей в текст.

Сохранение и компиляция

В конце нужно сохранить и скомпилировать программный модуль и указать для него идентификатор и наименование. В данном примере следует использовать название Discount Letter.

DRAFT

Что Еще Нужно Сделать

Хотя данный код будет работать, чтобы подготовить его, можно было применять в реальной жизни нужно сделать еще несколько веще:

- Не следует указывать название шаблона в коде. Оно должно храниться в таблице, а пользователь должен выбирать его в специальной форме. Возможно, потребуется несколько шаблонов для разных видов писем, которые будут отсылаться клиентам.
- Также нужно добавить обработку ошибок. Например, вызов функции CREATE будет неудачным, если на компьютере не установлен Word. Поэтому нужно выполнить проверку возвращаемого функцией CREATE значения и если ее вызов был неудачен, отобразить соответствующее сообщение.
- Если клиенту скидка предоставлять не будет, то пользователю должно выводиться сообщение. В данном примере программный модуль просто завершит свою работу.

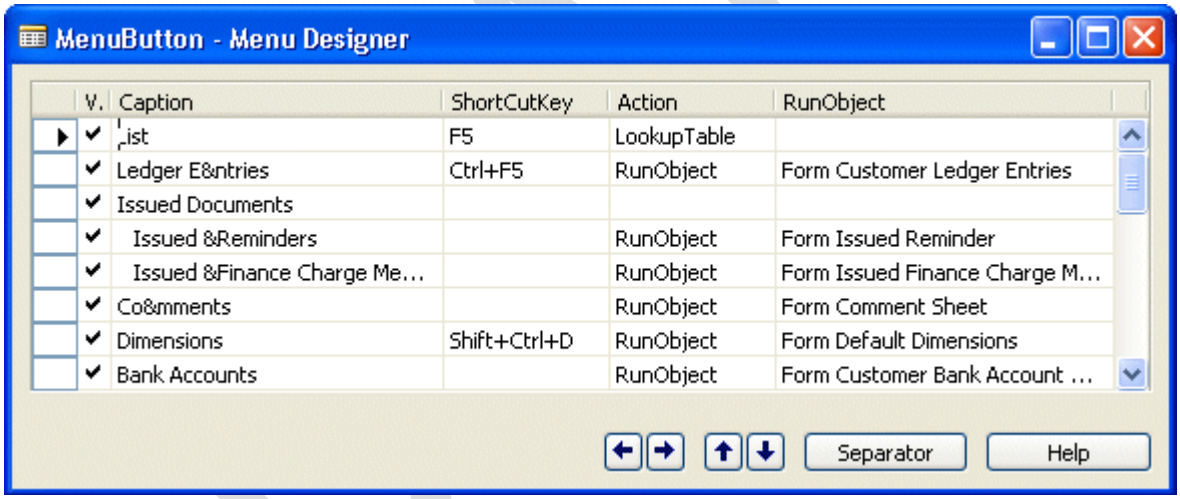
Вызов Программного Модуля из Карточки Клиента

Последней задачей будет вызов программного модуля из карточки клиента. Для этого нужно добавить элемент меню в кнопку Клиент (Customer).

1. Открыть **Object Designer** (SHIFT+F12) и щелкнуть кнопку **Form**.

2. Выбрать форму ***Customer Card*** (Form 21) и щелкнуть кнопку **Design**.

3. Щелкнуть правой кнопкой мыши по кнопке меню Клиент (Customer) и выбрать пункт Menu Items. Также для открытия **Menu Designer** можно щелкнуть Вид, Menu Items:



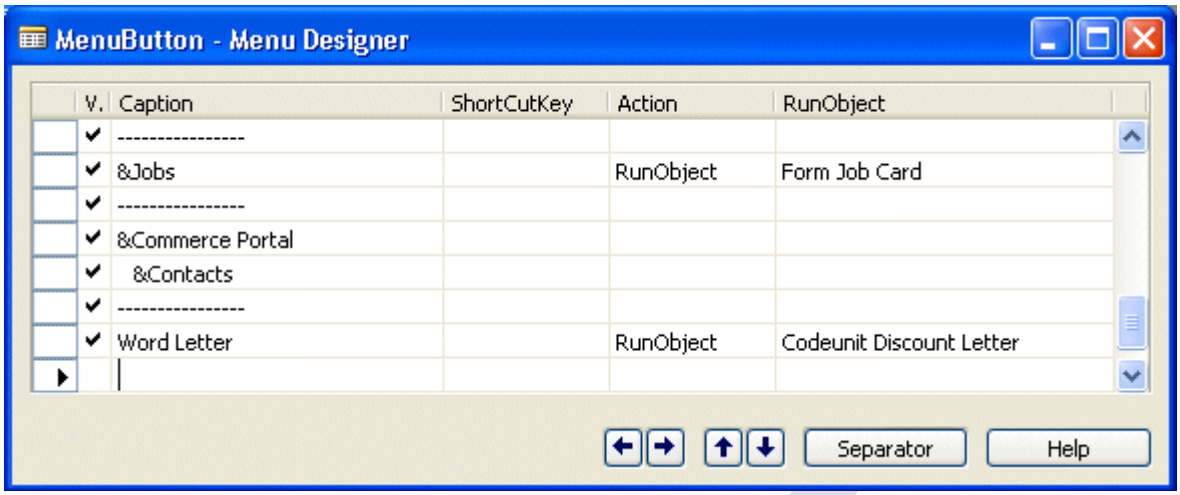
4. Прокрутите окно вниз, к концу списка элементов меню.

5. Щелкните кнопку Separator, чтобы вставить строку-разделитель.

6. Заполните поле **Caption** текстом, который должен отображаться в меню (введите *Word Letter*).

7. В поле **Action** щелкните кнопку AssistButton и выберите опцию *RunObject*.

8. В поле **RunObject** щелкните кнопку AssistButton и выберите из списка созданный ранее программный модуль:



9. Сохраните и откомпилируйте форму ***Customer Card***.

Созданное письмо содержит всего шесть полей и никакого текста. Перед тем как использовать письмо в реальной ситуации, возможно, потребует добавить еще поля, такие как название и адрес вашей компании, дату и код валюты, ну и конечно сам текст письма. Также нужно выполнить форматирование письма, чтобы сделать его привлекательным.

CRONUS International Ltd.
5 The Ring,
Westminster,
London,
GB

New Concepts Furniture
705 West Peachtree Street,
Atlanta,
US

Dear Ms. Tammy L. McDonald,

We are very pleased to see that our business relationship is thriving and that you have ordered goods for the value of GBP 222,241.32 from us. We are therefore delighted to be able to offer you a discount of 3% on any orders that you place with us during the rest of this calendar year.

Yours sincerely

Однако если вы измените, порядок полей в шаблоне, вы также должны изменить нумерацию полей в программном модуле, чтобы вставка данных осуществлялась в правильные поля.

Создание Диаграмм в Microsoft Excel

В данном примере будет выполнена передача данных из таблицы ***G/L Entry*** (Фин. Книга Операций) в Microsoft Excel и создана диаграмма. Основная цель данного примера показать, как работать с enumerations.

В данном примере будет создана диаграмма в Microsoft Excel, которая содержит информацию о расходах на продажу по отделам. В Плане Счетов эту информацию можно увидеть на счетах с 44-000 по 44-999. В таблице ***Dimension Value*** можно увидеть, что всего существует три отдела: *АДМ*, *ПРОДАЖИ* и *ПРОИЗ*. При выполнении примера будет создан программный модуль, который будет передавать данные из таблицы ***G/L Entry*** в Excel и создавать диаграмму. Данный программный

модуль будет запускаться непосредственно из Object Designer, но в настоящем приложении он должен будет запускаться из подходящего места, например, из меню в окне ***Chart of Accounts (План Счетов)***.

Создание Программного Модуля: Объявление Переменных

Первым шагом нужно определить в программном модуле необходимые переменные:

1. Откройте **Object Designer**, щелкните кнопку **Codeunit**, затем кнопку **New**, чтобы создать новый программный модуль.

2. Щелкните Вид, C/AL Globals.

3. Создайте переменную типа Record и Subtype = ***G/L Entry***. Назовите ее *GLEntry*.

4. Создайте переменную *xlApp* типа Automation, Automation Server = Microsoft Excel 11.0 Object Library, Automation Object = Application.

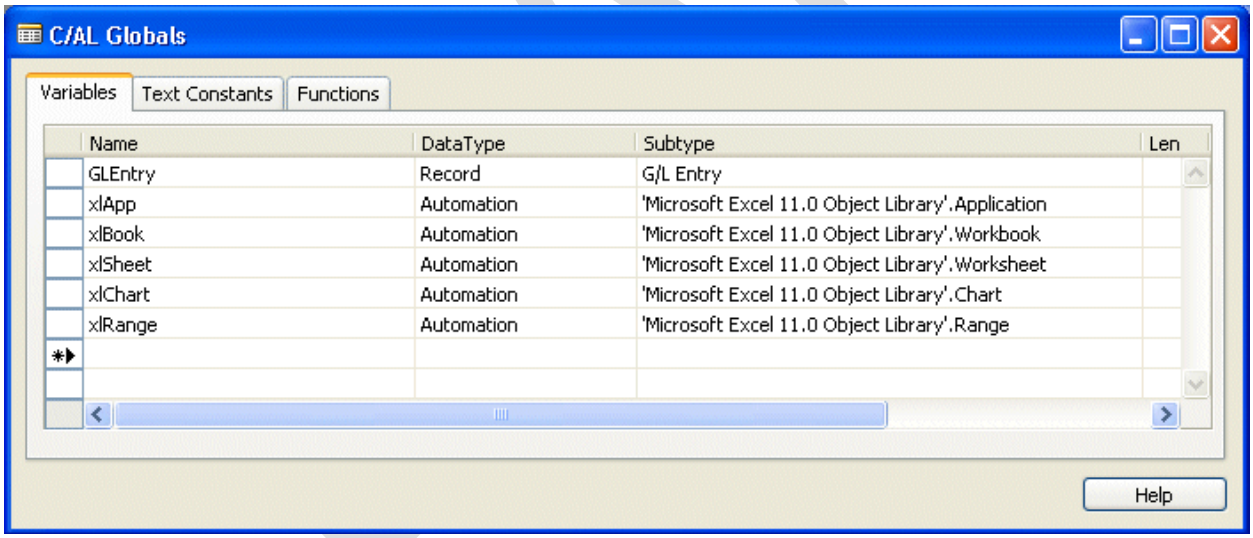
5. Создайте переменную *xlBook* типа Automation, Automation Server = Microsoft Excel 11.0 Object Library, Automation Object = Workbook.

6. Создайте переменную *xlSheet* типа Automation, Automation Server = Microsoft Excel 11.0 Object Library, Automation Object = Worksheet.

7. Создайте переменную *xlChart* типа Automation, Automation Server = Microsoft Excel 11.0 Object Library, Automation Object = Chart.

8. Создайте переменную *xlRange* типа Automation, Automation Server = Microsoft Excel 11.0 Object Library, Automation Object = Range.

В результате окно ***C/AL Globals*** должно выглядеть следующим образом:



Создание Программного Модуля: Подготовительные Действия

Сам по себе код достаточно простой. Во-первых, нужно установить походящий ключ для таблицы ***G/L Entry***, затем с помощью SETFILTER выбрать нужные финансовые счета:

```
GLEntry.SETCURRENTKEY("G/L Account No.","Business Unit Code",
  "Global Dimension 1 Code","Global Dimension 2 Code",
  "Close Income Statement Dim. ID","Posting Date");
GLEntry.SETFILTER("G/L Account No.",'44-000..44-999');
```

Далее создать экземпляр Excel:

```
CREATE(xlApp)
```

Теперь нужно добавить новую книгу в Excel:

```
xlBook := xlApp.Workbooks.Add(-4167);
xlSheet:= xlApp.ActiveSheet;
xlSheet.Name := 'Expenses';
```

В первой строке для создания новой книги используется метод Add из Workbooks collection. Далее используется свойство *ActiveSheet* класса Application, чтобы гарантировать, что все следующие действия будут выполняться на активном листе новой книге. В третьей строке листу задается имя.

Вероятно, вам интересно, что за параметр -4167 использовался в методе Add. Если посмотреть в онлайновой справке Microsoft Excel Visual Basic, то можно увидеть, что в метод Add объекта Workbooks передается один параметр - Template. Его тип данных VARIANT.

В описании указано:

Если данный параметр содержит константу, то новая книга будет содержать один специальный лист. Значение может быть типа XlWBATemplate: xlWBATChart, xlWBATExcel4IntlMacroSheet, xlWBATExcel4MacroSheet, или xlWBATWorkSheet.

Нам требуется создать рабочую книгу с одни листом. Согласно описанию нужно в качестве параметра Template передать константу XlWBATemplate со значением xlWBATWorkSheet.

Вместо этого была передано число -4167. Далее будет рассказано почему.

Enumerations

Как было сказано в разделе «Использование Технологий COM в Среде C/SIDE» в этом случае тип VARIANT является enumeration.

Существует для вида enumerations: являющиеся USERDEF и не являющиеся. В данном случае это не USERDEF потому, что в C/AL Symbol Menu он отображается как VARIANT. Чтобы убедится, что это enumeration нужно обратиться к справке по Microsoft Excel Visual Basic. Совет: если название параметра начинается с xl*, то он, скорее всего, является константой (в Microsoft Word это будет wd*, а в Microsoft Outlook - ol*).

Среда C/SIDE не позволяет использовать символьные названия (xlWBATWorkSheet). Поэтому нужно использовать enumerator (-4167). Ниже будет рассказано, как найти это значение.

Поиск значения enumerator:

1. Открыть Excel.

2. Щелкнуть Tools, Macro, Visual Basic Editor.

3. Щелкнуть View, Object Browser.

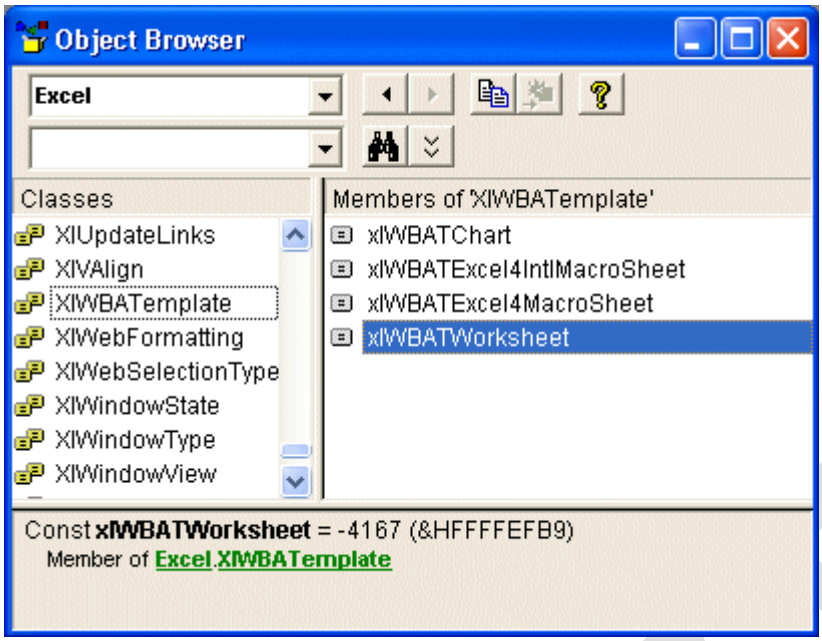
4. Выбрать *Excel* в выпадающем списке в верхнем левом углу формы.

5. В списке классов (Classes) прокрутить до XlWBATemplate и выбрать его.

6. В списке членов класса XlWBATemplate (Members of 'XlWBATemplate') выбрать xlWBATWorkSheet.

7. Значение будет показано в информационной панели в нижней части формы:

DRAFT



Как вариант можно использовать следующий метод:

Создать в Excel макрос, который вызывает функцию MsgBox:

```
Sub x()
  MsgBox (xlWBATWorksheet)
End Sub
```

При запуске макроса появится окно, в котором будет отображено значение enumerator:

Создание Программного Модуля: Передача Данных

Чтобы передать данные нужно выполнить две вещи: рассчитать данные и передать результаты расчета. Для расчета данных нужно сделать следующее:

```
GLEntry.SETRANGE("Global Dimension 1 Code", 'АДМ');
GLEntry.CALCSUMS(Amount);
```

Функция SETRANGE используется, чтобы установит фильтр на записи таблицы ***G/L Entry*** по полю **Global Dimension 1 Code**. АДМ (Администрация) – это первый отдел. Затем нужно использовать функцию CALCSUMS(Amount), чтобы рассчитать сумму отдела АДМ.

Теперь можно передать данные в Microsoft Excel:

```
xlSheet.Range('A2').Value := 'Administration';
xlSheet.Range('A3').Value := GLEntry.Amount;
```

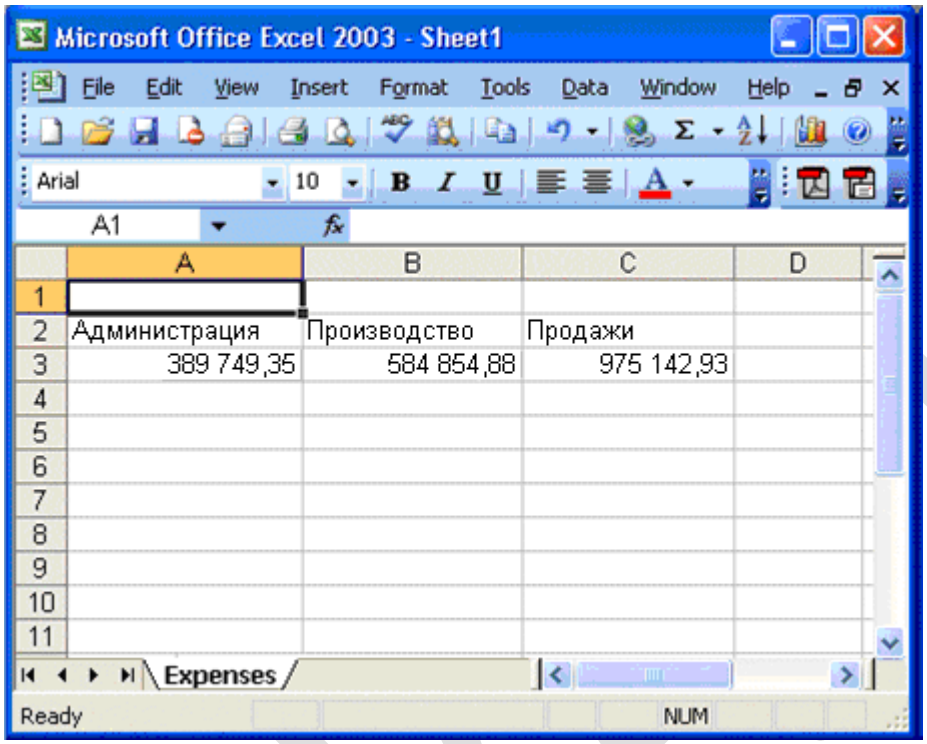
Эти же действия нужно повторить для двух оставшихся отделов ПРОДАЖИ и ПРОИЗ:

```
GLEntry.SETRANGE("Global Dimension 1 Code", 'ПРОИЗ');
GLEntry.CALCSUMS(Amount);
xlSheet.Range('B2').Value := 'Production';
```

DRAFT

```
xlSheet.Range('B3').Value := GLEntry.Amount;
GLEntry.SETRANGE("Global Dimension 1 Code", 'ПРОДАЖИ');
GLEntry.CALCSUMS(Amount);
xlSheet.Range('C2').Value := 'Sales';
xlSheet.Range('C3').Value := GLEntry.Amount;
```

Данные, переданные в Microsoft Excel, будут выглядеть следующим образом:



Создание Программного Модуля: Создание Диаграммы

Последним шагом будет создание диаграммы. Для этого воспользуемся методом ChartWizard, а диаграмму выберем объемную круговую диаграмму (3D pie). Это быстрый и простой способ создания диаграмм. Более тонко контролировать созданиее диаграммы можно, используя методы и свойства различных объектов Chart (ChartArea, Legend, и т.д.).

Первым делом нужно задать диапазон данных для диаграммы:

```
xlRange := xlSheet.Range('A2:C3')
```

Затем нужно добавить новый лист и задать ему имя:

```
xlChart := xlBook.Charts.Add
xlChart.Name := 'Expenses - Graph'
```

И на последнем этапе нужно создать диаграмму:

```
xlChart.ChartWizard(xlRange,4102,7,1,1,0,0,'Expenses');
```

При этом будут использованы первые восемь необязательных параметров метода ChartWizard:

Параметр	Описание	Значение
Source	Диапазон, в котором содержатся данные для новой диаграммы	Объект xlRange возвращен методом xlSheet.Range('A2:C3').
Gallery	Тип диаграммы	-4102. enumerator для xl3DPie из xlChartType enumeration.

Параметр	Описание	Значение
Format	Необязательный параметр, встроенные форматы	7 – это значение найдено экспериментальным путем, т.к. в документации к Excel об этом параметре мало информации.
PlotBy	Указываются: данные в строках или по столбцам	1. enumerator для xlRows из XlRowCol enumerator.
CategoryLabels	Целое число, указывает количество строк (столбцов) внутри диапазона данных, где содержатся подписи к данным.	1 – одна строка для надписей (названия отделов).
SeriesLabels	Целое число, указывает количество строк (столбцов) внутри диапазона данных, где содержатся подписи к сериям	0 – в данных нет подписей к сериям.
HasLegend	TRUE чтобы включить легенду	0 – данное значение работает.
Title	VARIANT с названием	Вы передали строку, 'Expenses'.

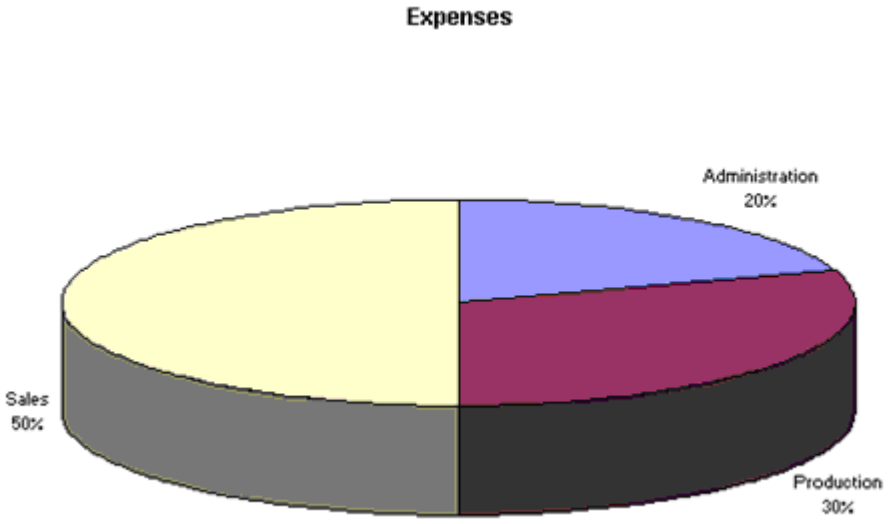
И в конце нужно сделать Excel видимым:

```
xlApp.Visible := TRUE
```

С Excel может произойти ошибка (General Protection Fault) при закрытии нового листа, созданного, пока Excel был невидимым. Чтобы решить эту проблему можно сделать Excel видимым сразу после создания нового листа. Как вариант, можно сделать Excel видимым перед созданием нового листа и сразу же после создания листа сделать его вновь невидимым. Для этого нужно написать:

```
xlApp.Visible := TRUE;
xlBook := xlApp.Workbooks.Open(FileName);
xlApp.Visible := FALSE;
```

Диаграмма будет выглядеть следующим образом:



DRAFT

19.4 Получение Событий в C/SIDE

Среда C/SIDE может получать события от automation servers и OCX, которыми она управляет. При объявлении глобальной переменной с типом Automation можно указать, что требуется получить события. Для этого нужно установить в свойстве *WithEvents* значение *Yes*. В этом случае будут автоматически созданы триггеры C/AL для событий данного Automation. Название триггера состоит из названия переменной, за которым следует "::<Event name>". Например, если объявить переменную с именем MyEventVar, а название события MessageReceived(...), то название триггера будет MyEventVar::MessageReceived(...).

В следующем примере будет показано, как получить в среде C/SIDE события от Dynamics NAV Communication Component.

Для дополнительной информации о Dynamics NAV Communication Component и способе, которым он управляет обменом потоками данных между NAV Application Server и bus adapter, см в онлайновой справке *Development Guide for Communication Components* (devguide.chm). Данный файл обычно находится в C:\Program Files\Common Files\Dynamics NAV\Communication Component.

Dynamics NAV Communication Component и bus adapter устанавливаются как часть NAV Application Server. Они должны быть установлены, перед тем как вы начнете выполнять следующий пример.

Получение Уведомления о Входящем Документе

Перед продолжением будет полезно ознакомиться с примером «Создания Письма в Microsoft Word».

В следующем примере будут созданы два программных модуля: один позволит среде C/SIDE получать события – уведомления о входящем потоке, а второй позволит клиенту отправлять поток. В реальной жизни один и тот же клиент не будет сам себе отправлять и получать потоки. Чтобы отправлять и получать потоки в среде C/SIDE, требуется установить Dynamics NAV Communication Component и socket bus adapter.

Получение События

Выполнение примера мы начнем с создания программного модуля, который позволяет клиенту получать события.

Объявление переменных для внешних компонент

Первым шагом будет создание программного модуля и объявления нескольких переменных типа Automation для обоих компонент. Для каждой переменной нужно выбрать automation server и класс.

Объявите следующие переменные:

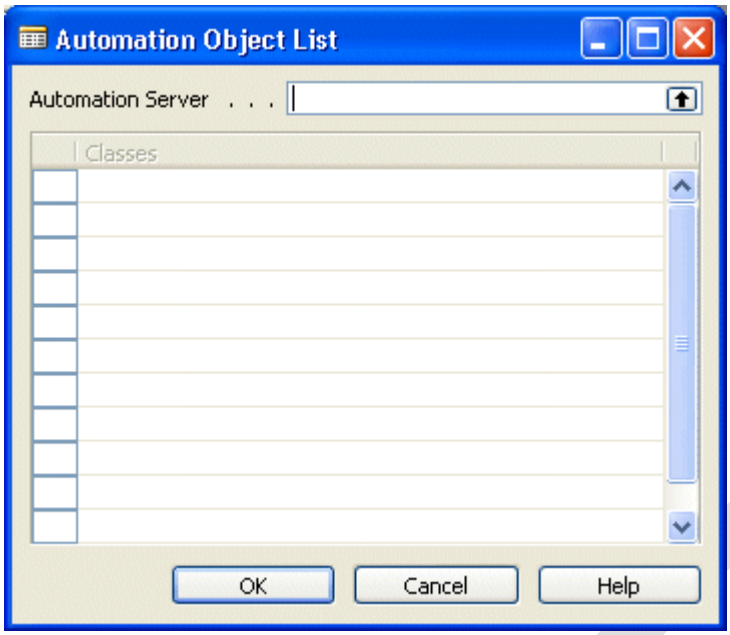
Variable Name	Data Type	Subtype
CC2	Automation	Dynamics NAV Communication Component version 2.CommunicationComponent
SBA	Automation	Dynamics NAV Socket Bus Adaptor.SocketBusAdapter
InMsg	Automation	Dynamics NAV Communication Component version 2.In Message
InS	InStream	
txt	Text	

Чтобы объявить данные переменные:

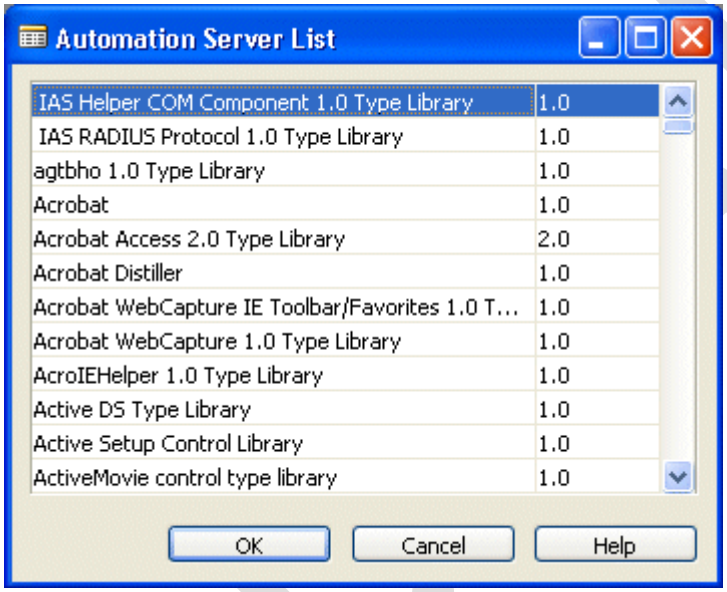
1. Щелкните Вид, C/AL Globals, чтобы открыть окно ***C/AL Globals***.

2. Введите *CC2* в качестве названия первой переменной и выберите тип данных Automation.

3. В поле **Subtype** щелкните кнопку AssistButton, чтобы открыть окно ***Automation Object List***.

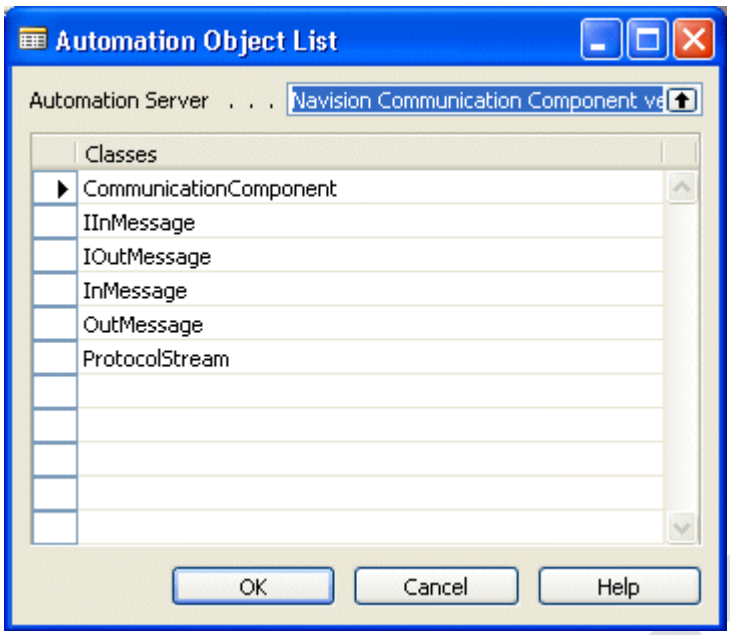


4 В поле **Automation Server** щелкните кнопку AssistButton, чтобы открыть окно ***Automation Server List***.



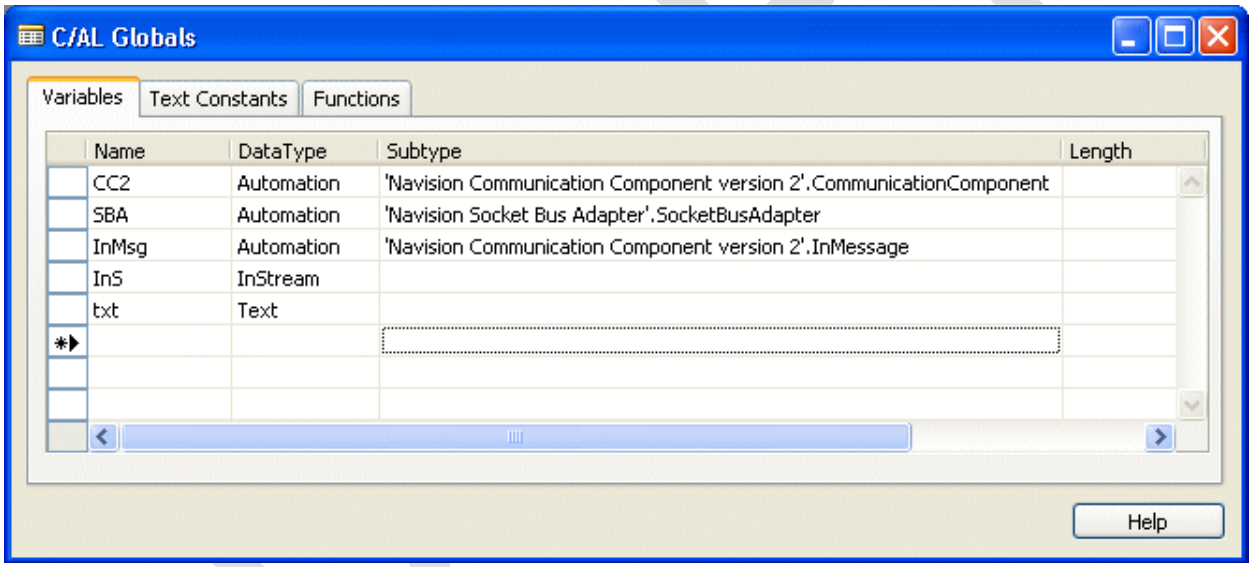
5 Выберите Dynamics NAV Communication Component version 2 и щелкните кнопку OK.

6 Теперь в окне ***Automation Object List*** отображаются классы Dynamics NAV Communication Component.



7. Выберите класс CommunicationComponent.

8. Объявите остальные переменные, указанные ранее, чтобы окно ***C/AL Globals*** выглядело следующим образом:



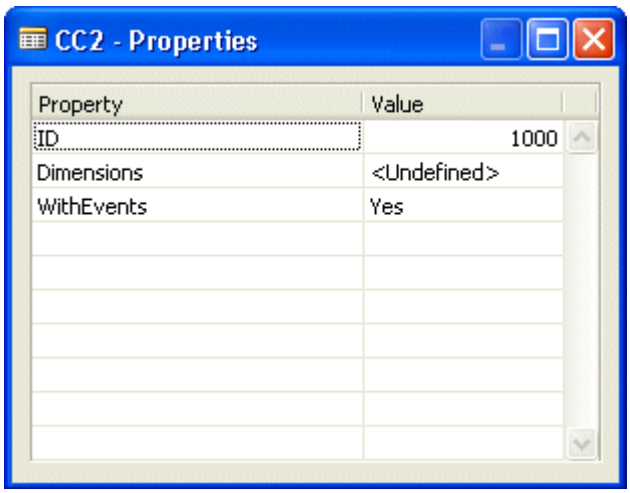
Настройка свойства *WithEvents*

Теперь, когда объявлены переменные для Dynamics NAV Communication Component и bus adapter, нужно установить в свойстве *WithEvents* для Dynamics NAV Communication Component значение *Да*. Сделав это, вы активизируете обработку событий от Dynamics NAV Communication Component. Это значит, что C/SIDE может получать уведомления о любых входящих потоках.

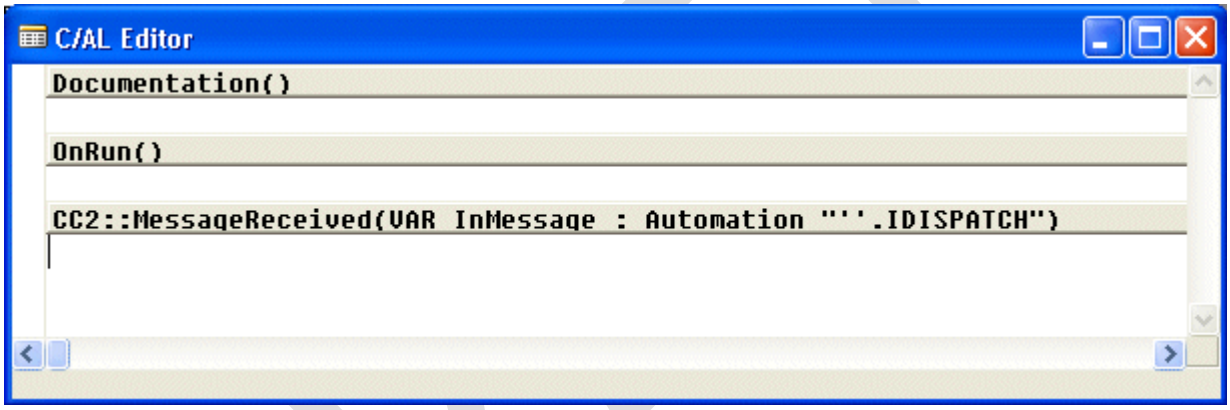
Чтобы настроить свойство *WithEvents* нужно выполнить следующие шаги:

1. Откройте окно ***C/AL Globals*** и выберите, созданную ранее переменную CC2.

2. Откройте окно ***Properties*** и установите для свойства *WithEvents* значение *Да*:



После установки в свойстве *WithEvents* значения *Да* в программном модуле будут автоматически созданы триггеры для каждого события, которое может быть создано выбранной глобальной переменной. Для Dynamics NAV Communication Component будет создан следующий триггер для события:



Создание automation servers

Перед тем как можно будет использовать Dynamics NAV Communication Component и bus adapter нужно создать экземпляры каждого из них. Для этого нужно использовать функцию C/AL CREATE. Для вызова CREATE введите в триггере OnRun программного модуля следующий код:

```
OnRun()
IF ISCLEAR(CC2) THEN
    CREATE(CC2);
  IF ISCLEAR(SBA) THEN
    CREATE(SBA);

  CC2.AddBusAdapter(SBA, 0);
  SBA.OpenSocket(8079, '');
```

Данный код создает экземпляры Dynamics NAV Communication Component и bus adapter. Bus adapter теперь использует порт 8079 на локальном компьютере.

Написание кода в триггере

Следующим шагом будет написание кода в триггере MessageReceived, который отслеживает возникновения события. Введите в триггер MessageReceived следующий код:

```
CC2::MessageReceived(VAR InMessage : Automation "'.IDISPATCH")
InMsg:= InMessage;

 InS:= InMsg.GetStream();

 WHILE NOT (InS.EOS) DO
 BEGIN
   InS.READ(txt);
   MESSAGE(txt);
 END;

 InMsg.CommitMessage();
```

Данный код получает переданную строку, и с помощью функции InStream.EOS определяет, достигнут ли End of Stream (EOS). Для чтения из объекта InStream используется функция InStream.READ. Чтение данных выполняется в двоичном формате. В конце отображается текст, полученный во входящем сообщении.

У функции InStream.READ есть необязательный параметр:

InStream.Read(Variable, [Length])

Этот параметр позволяет указать число байт, которое нужно прочитать из объекта InStream. Дополнительную информацию о потоках (streams) смотрите в онлайновой справке *C/SIDE Reference Guide*.

> **Важно**
>
> Если удалить глобальную переменную, с которой связаны триггеры событий, то и триггеры и содержащийся в них код будут удалены. Более того, если изменить Data Type или Subtype глобальной переменной, также будут удалены триггеры событий и содержащийся в них код.

Сохраните и откомпилируйте программный модуль.

Триггеры Событий

Если вы включили для среды C/SIDE получение событий от компонент, которые в ней содержатся, нужно знать о следующих ограничениях. Эта информация также касается разработчиков компонент.

Для триггеров, автоматически создаваемых для событий, существуют некоторые ограничения. Также есть ограничения для получаемых этими триггерами данных.

Ограничения для Триггеров

- Среда C/SIDE поддерживает только исходящий интерфейс по умолчанию, определенный для переменной automation. Если для класса определено несколько исходящих интерфейсов, триггер события будет автоматически создан только для исходящего интерфейса по умолчанию.
- В вызываемую функцию можно передать максимум 39 параметров.
- Передаваемая строка может содержать максимум 1024 символа.
- В COM используется стратегия стыкуемого объекта для связи Dynamics NAV и automation server. Определенный в данной стратегии объект Sink, внедренный в Dynamics NAV поддерживает только интерфейс IDispatch (и IUnknown). Поэтому ожидается, что automation server будет использовать IDispatch при выполнении событий.
- Название параметров обрезается до 30 символов.

DRAFT

- Для триггеров событий нет возвращаемых значений.
- Название триггера, состоящее из названия переменной, "::" и названия события обрезается до 30 символов.

Ограничения для Входящих Данных

Все получаемые данные копируются во внутренний тип данных, который может обрабатывать любой тип данных COM. В процессе данного преобразования никакие данные не теряются, проверка на корректность относительно типов данных C/AL. Данные остаются в этом внутреннем типе данных пока они не будут использованы в триггере. Когда данные используются, они преобразуются в соответствующий тип данных C/AL.

Однако если тип данных Variant, то никаких преобразований не выполняется. При преобразовании данные не теряются, выполняются все необходимые проверки.

Если преобразование не может быть выполнено, например, из-за некорректного типа данных, триггер события отобразит сообщение об ошибке и завершит свою работу. Обратите внимание, если данные не будут использоваться в триггере, то не будут выполняться и проверки на корректность типов данных.

Если параметр типа VAR (называется, ByRef) и данный используются в триггере события, то выполняется скрытое преобразование прямо тем, как триггер вернет значение. Если преобразование не может быть выполнено, будет показано сообщение об ошибке и триггер завершит свою работу.

Отправка Потока

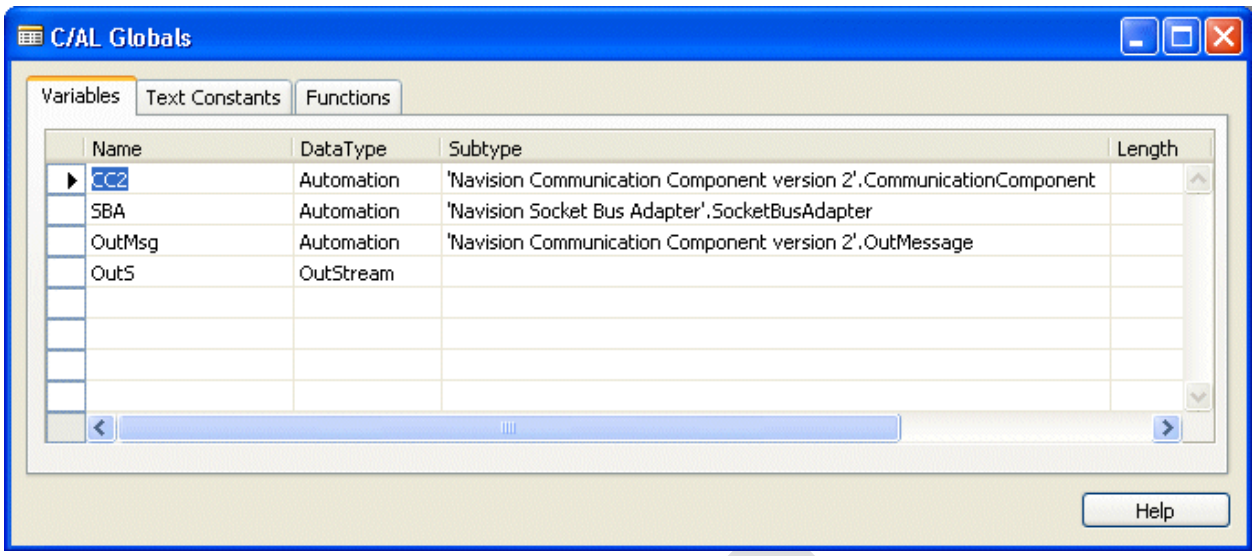
На следующем шаге данного примера будет создан программный модуль, отправляющий поток.

1. Начнем с создания нового программного модуля и объявления следующих переменных:

Variable Name	Data Type	Subtype
CC2	Automation	Dynamics NAV Communication Component version 2.CommunicationComponent
SBA	Automation	Dynamics NAV Socket Bus Adaptor.SocketBusAdapter
OutMsg	Automation	Dynamics NAV Communication Component version 2.OutMessage
Outs	InStream	

Окно ***C/AL Globals*** должно выглядеть следующим образом:

DRAFT



Отправка сообщений

2. Добавьте следующий код в триггер OnRun программного модуля:

```
OnRun()
IF ISCLEAR(CC2) THEN
    CREATE(CC2);
  IF ISCLEAR(SBA) THEN
    CREATE(SBA);

  CC2.AddBusAdapter(SBA, 0);
  OutMsg:= CC2.CreateoutMessage('Sockets://localhost:8079');

  OutS:= OutMsg.GetStream();
  OutS.WRITE('Hello world');
  OutS.WRITE('Using my first');
  OutS.WRITE('Navision Socket bus adapter');
```

Этот код создает экземпляр Dynamics NAV Communication Component и bus adapter. Затем создается исходящее сообщение (outmessage). Если исходящее сообщение не может быть создано, то возвращается NULL.

Функция ICommunicationComponent::CreateOutMessage требует параметр. Синтаксис параметра:

Bus adapter type://destination

Например:

Named pipe://myServer/myPipe

Sockets://myserver::portnumber

Message queue://MyMessageQueueServer\MyQueue

Socket bus adapter используется для отправки исходящего сообщения на порт 8097.

Затем определяется текст исходящего сообщения.

3. Сохранить и компилировать программный модуль.

Запуск и Тестирование Примера

Последним шагом будет тестирование программных модулей созданных при выполнении данного примера.

1. Запустите клиент Dynamics NAV и откройте базу данных, содержащую созданные программные модули.

2. Выберите в Object Designer программный модуль, который получает поток, и щелкните кнопку Run. Это программный модуль со свойством Single Instance = Да, поэтому он будет загружен в память и останется в ней.

3. Запустите еще одни клиент Dynamics NAV и откройте ту же базу данных.

4. В Object Designer выберите программный модуль, который отправляет поток, и щелкните кнопку Run.

5. Вернитесь к первому клиенту, в нем вы должны увидеть, что сообщение получено и отображается на экране.

19.5 Использование компонентов (Custom Controls) из Среды C/SIDE

Custom Controls также известны как компоненты OLE или компоненты ActiveX. Также они называются OCX потому, что файлы часто имеют расширение .ocx. Если нужно использовать компонент OCX, то требуется определить переменную (глобальную или локальную) типа *OCX* и указать для этой переменной соответствующий subtype.

Простой Пример

Продемонстрируем использования компонентов OCX на примере C/FRONT.OCX. Вместе с C/FRONT поставляется файл справки, в котором содержится описание его методов и свойств.

В данном примере будет создана простая форма, которая использует C/FRONT OCX для подключения к серверу, открытия базы данных и фирмы, и подсчитывает количество записей в указанной таблице. Эта форма будет иметь доступ к функциям и свойствам предоставляемым C/FRONT OCX.

Установка и Регистрация Компонента

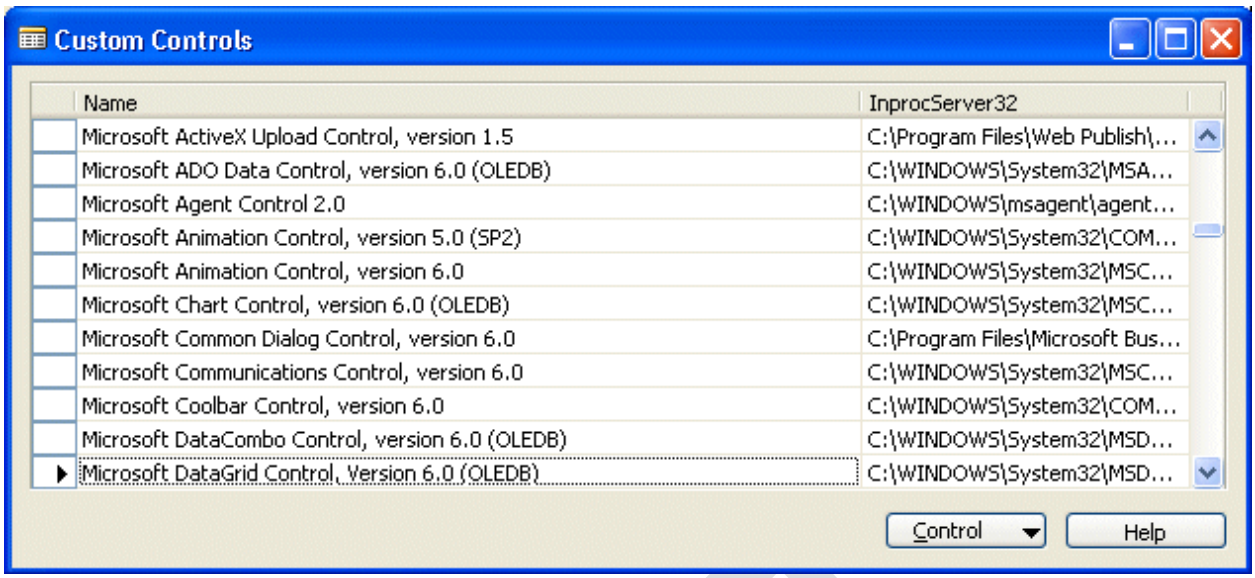
Перед тем как компонент можно будет использовать он должен быть физически установлен на компьютер. Кроме того он должен быть зарегистрирован в операционной системе.

Если компонент физически скопирован на жесткий диск, но еще не зарегистрирован, то вы можете выполнить процедуру регистрации компонента прямо из C/SIDE:

1. Скопируйте компонент с носителя на жесткий диск. Каталог C/FRONT содержит файл readme.txt с более детальной информацией.

2. Щелкните Сервис, Компоненты OCX, чтобы открыть окно ***Компоненты OCX***:

DRAFT



3. Щелкните кнопку Компонент, Просмотр, чтобы открыть стандартное диалоговое окно Windows, с помощью которого вы найдете требующий регистрации компонент OCX.

4. Когда компонент найден, выберите его и щелкните кнопку Open.

Теперь компонент зарегистрирован в системе. Система отобразит сообщение, информирующее вас, что регистрация выполнена. Щелкните кнопку OK, чтобы вернуться в окно **Компоненты OCX**. Зарегистрированный компонент появится в списке.

Использование Компонента из C/AL

Перед тем как вы сможете использовать методы и свойства компонента, он должен быть предварительно объявлен как глобальная или как локальная переменная. Сделав это, вы сможете использовать его методы и свойства, а также видеть эти методы и свойства в окне Symbol Menu. Если в Symbol Menu выделить метод или свойство и нажать клавишу F1, то будет показана контекстная справка по этому методу или свойству (предполагается, что файл справки поставляется поставщиком компонент). C/FRONT поставляется с файлом справки cfont.chm.

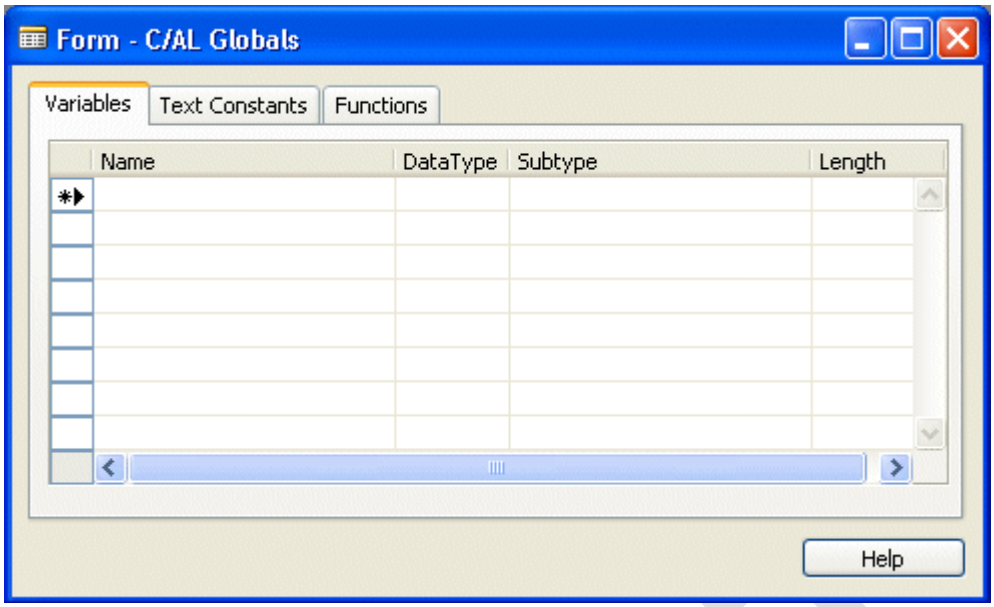
Объявление Компонентов как Переменных

Чтобы объявить компонент в качестве переменной:

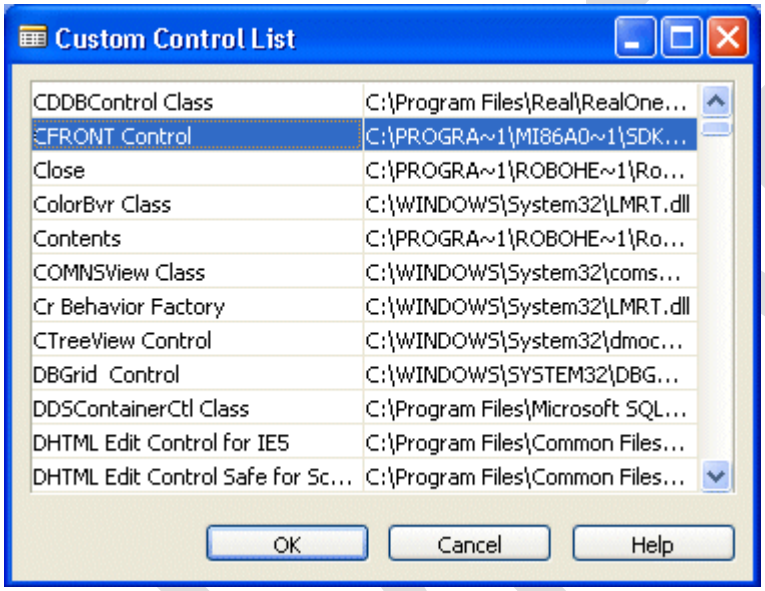
1. В окне Object Designer создайте новую пустую форму, которая не связана ни с одной таблицей.

2. Щелкните Вид, C/AL Globals, чтобы открыть следующее окно:

DRAFT

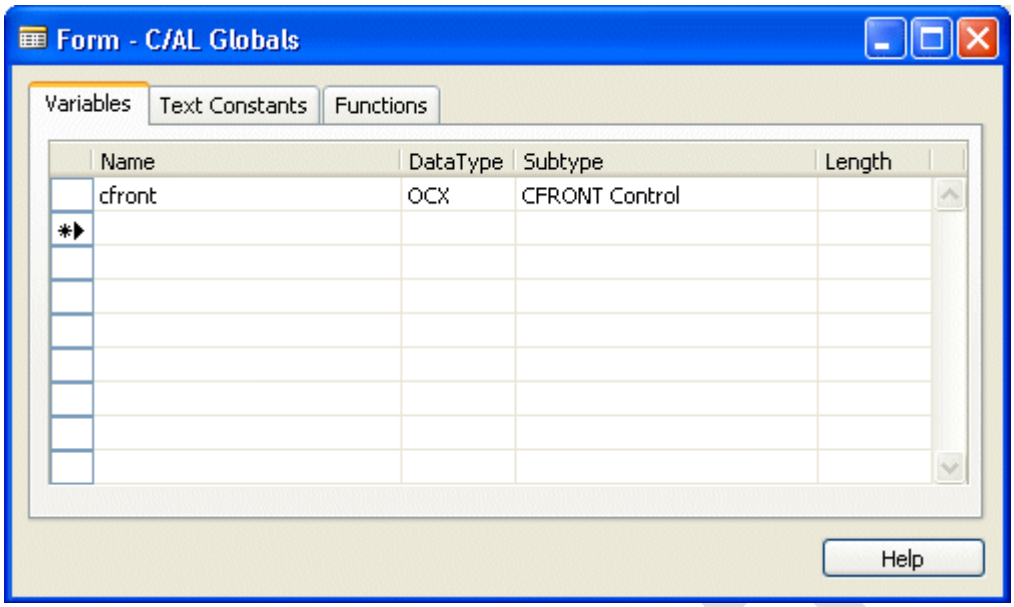


3. Укажите имя переменной (в данном примере, *cfront*), и выберите тип данных OCX. В поле **Subtype** щелкните кнопку AssistButton, чтобы открыть следующее окно:



4. Найдите в списке компонент, выделите его и щелкните кнопку OK.

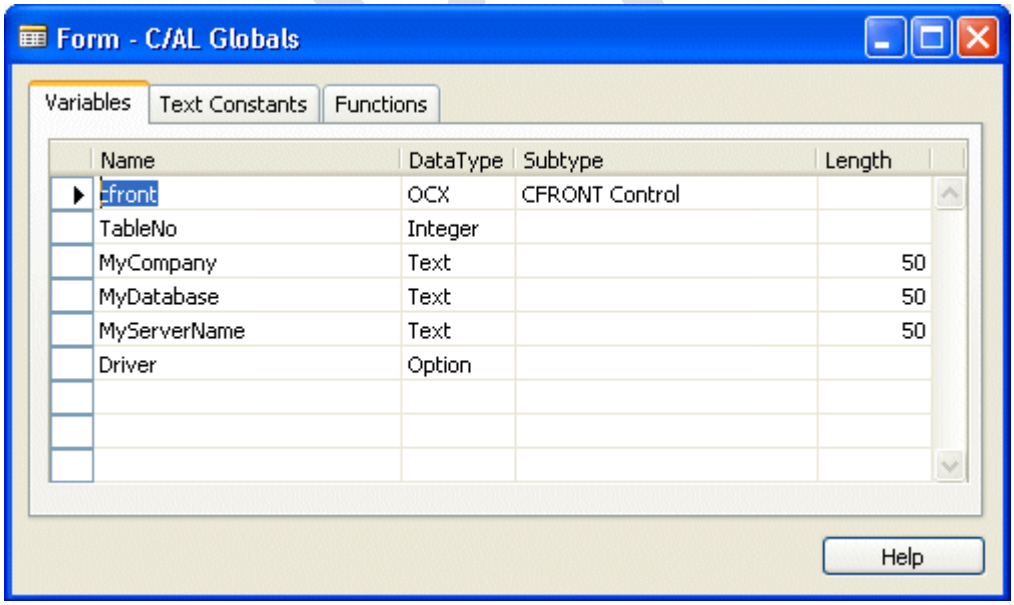
5. Когда вы уберете курсор из поля **Subtype,** в нем появится название компонента, и окно будет выглядеть следующим образом:



6. Объявите следующие переменные:

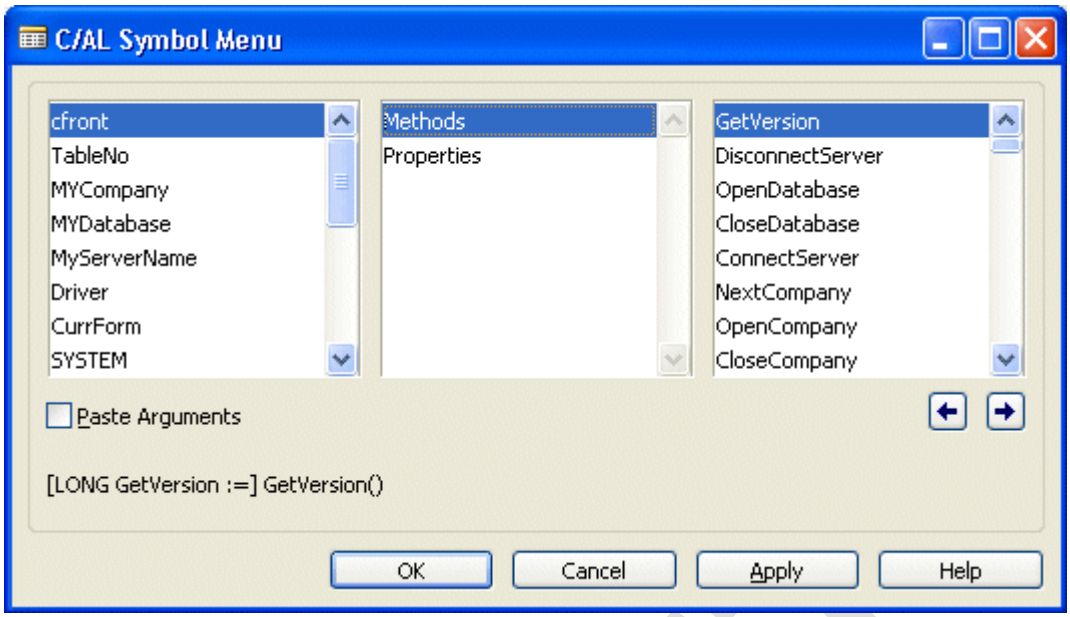
Name	DataType	Subtype	Length
TableNo	Integer		
MyCompany	Text		50
MyDatabase	Text		50
MyServerName	Text		50
Driver	Option		

7. Теперь окно ***C/AL Globals*** должно выглядеть следующим образом:



8. Выберите переменную с именем Driver и откройте для нее окно ***Properties*** (Shift+F4). В свойстве *OptionString* введите *NDBCS,NDBCN*.

Все переменные добавлены в C/AL Symbol Menu, и теперь вы имеете доступ ко всем методам и свойствам C/FRONT OCX. Чтобы проверить это щелкните Вид, C/AL Symbol Menu (F5). Откроется окно ***C/AL Symbol Menu***:

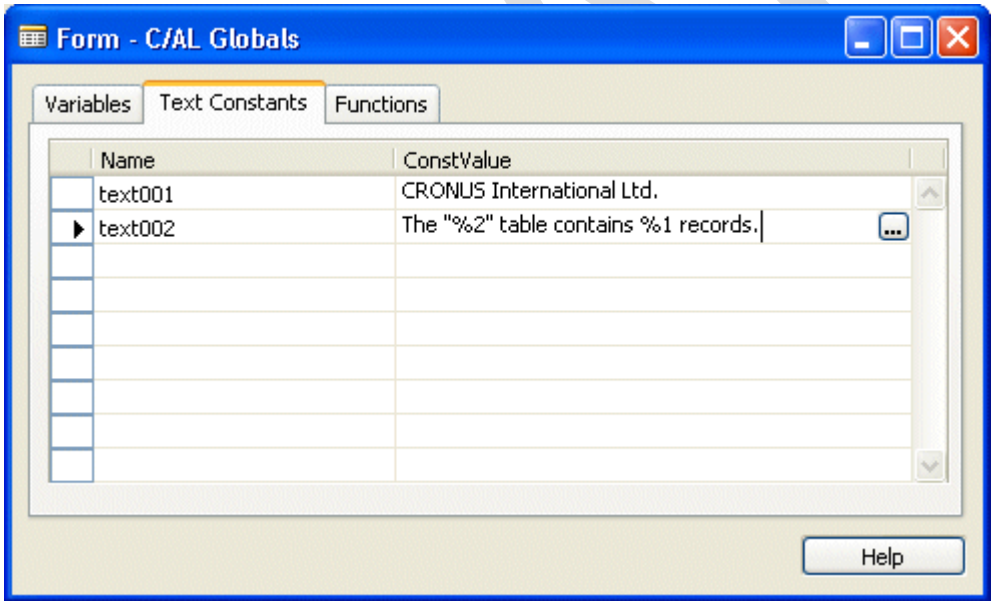


Как видите, методы и свойства C/FRONT теперь доступны. Выберите один из методов или свойство в правом столбце и нажмите клавишу F1. Откроется онлайновая справка по C/FRONT и отобразит описание выбранного метода или свойства.

Добавление текстовых констант

Следующим шагом добавим текстовые константы, которые используются данной формой.

Откройте окно ***C/AL Globals*** и создайте следующие текстовые константы:



Создание Формы

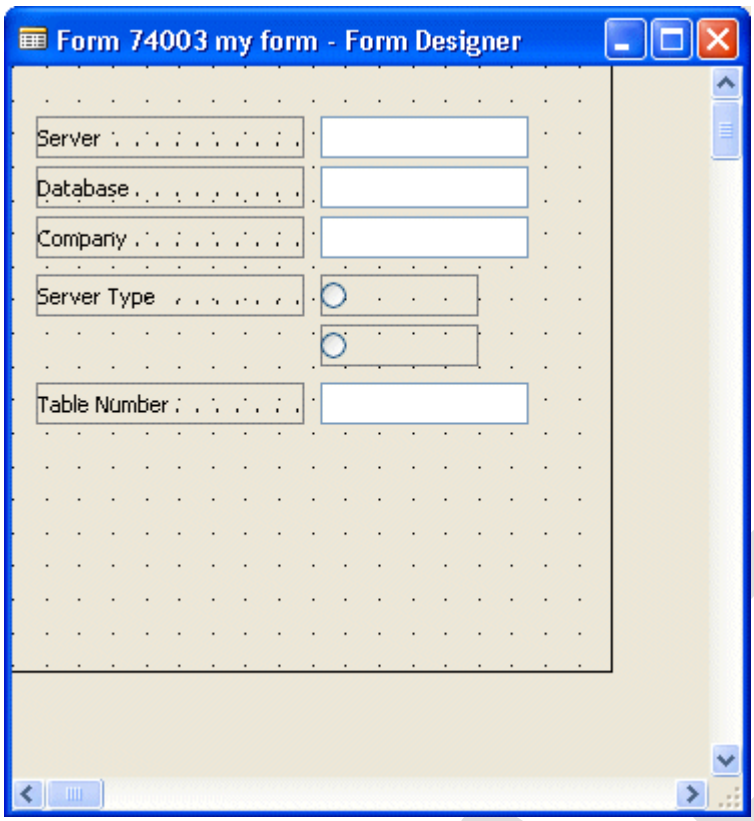
Первым шагом при создании формы является добавление необходимых полей.

Добавление полей

1. Откройте панель инструментов **Toolbox** и добавьте четыре текстовых зоны и две кнопки выбора на форму.

2. Добавьте надписи для текстовых зон и первой кнопки выбора: *Server*, *Database*, *Company, Table Number* и *Server Type*, соответственно.

3. Когда вы закончите форма должна выглядеть следующим образом:



Вы можете передвинуть поля необходимым образом, чтобы форма соответствовала требуемому виду.

4. Выберите текстовую зону Server и щелкните Вид, Properties (Shift+F4), чтобы открыть окно ***Properties***.

5. В свойстве *SourceExpr* щелкните кнопку AssistButton и откройте ***C/AL Symbol Menu***.

6. Выберите *MyServerName* и убедитесь, что флаг в поле **Paste Arguments** установлен. Затем щелкните кнопку OK.

Текстовая зона на форме теперь должна содержать следующий текст:

<=MyServerName>

7. Повторите эту процедуру для остальных трех текстовых зон, добавив переменные MyDatabase, MyCompany и TableNo в соответствующие текстовые зоны.

8. Выберите первую кнопку выбора и в окне ***Properties*** введите следующие значения:

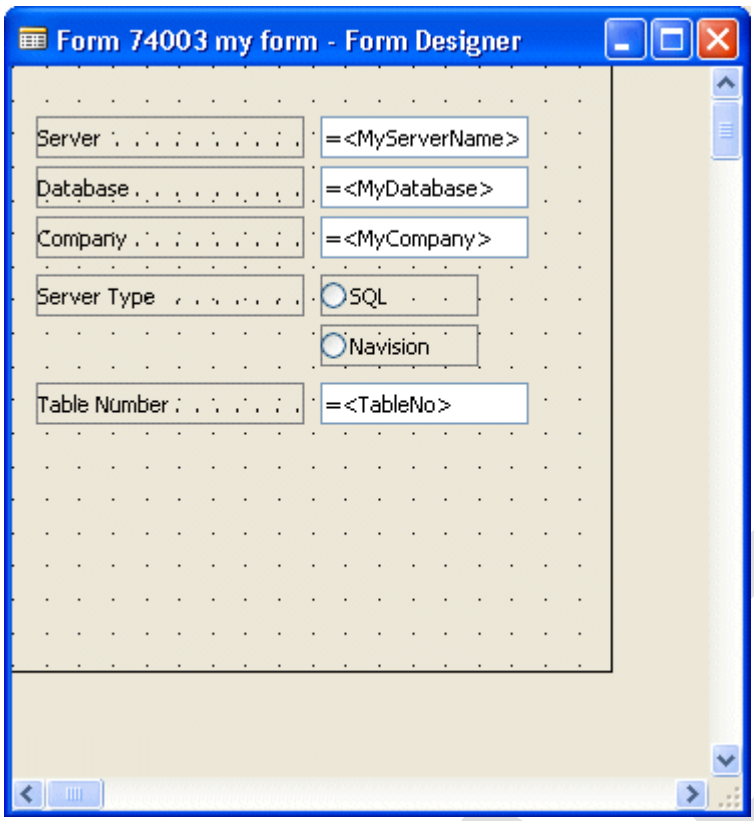
Property	Value
Caption	SQL
SourceExpr	Driver
OptionValue	NDBCS

9. Выберите вторую кнопку выбора и в окне ***Properties*** введите следующие значения:

Property	Value
Caption	Dynamics NAV
SourceExpr	Driver
OptionValue	NDBCN

Теперь форма должна выглядеть следующим образом:

DRAFT



Добавление Кода на Форму

Теперь, когда создана основная форма, наступило время добавить в нее код, чтобы заставить форму работать.

В данном примере большая часть кода будет помещена в триггер *OnPush* командной кнопки, расположенной в нижней части формы.

1. Откройте форму в **Form Designer** и добавьте на нее командную кнопку.

2. Выберите командную кнопку и откройте окно ***Properties*** (SHIFT+F4) и введите *Connect* в качестве заголовка (caption).

3. Выберите командную кнопку и щелкните Вид, C/AL Code (F9) чтобы открыть C/AL Editor.

4. Введите следующий код в триггер *OnPush*:

```
CLEAR (cfront);
cfront.ConnectServerAndOpenDatabase(FORMAT(Driver),
MyServerName,'tcp ', MyDatabase,0,FALSE,TRUE,'','');
cfront.OpenCompany(MyCompany);
IF cfront.OpenTable(TableNo,TableNo) THEN
  MESSAGE (text002, cfront.RecCount (TableNo),cfront.TableName(TableNo));
cfront.CloseTable(TableNo);
MESSAGE ('Table closed');
cfront.CloseCompany();
MESSAGE ('Company Closed');
cfront.DisconnectServer();
MESSAGE ('Disconnect')
```

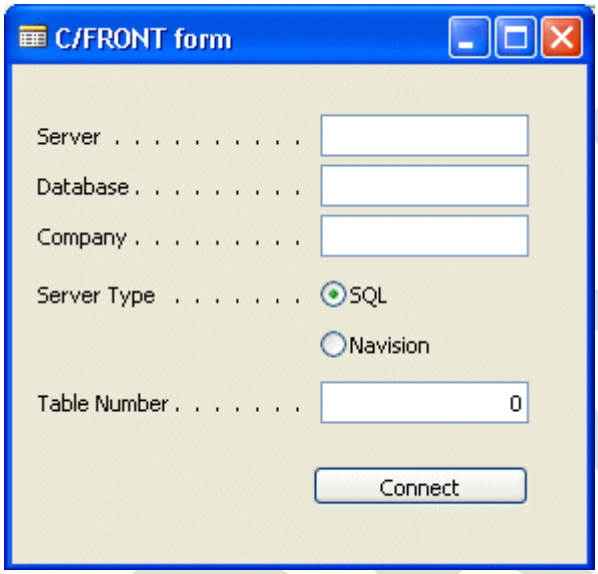
DRAFT

Для ввода функций C/FRONT откройте окно ***C/AL Symbol Menu*** и выберите требуемый метод или свойство. Убедитесь, что в поле **Paste Arguments** установлен флаг и щелкните кнопку OK. Функция C/FRONT будет вставлена в C/AL Editor и все, что от вас потребуется это лишь исправить параметр.

Этот код использует функции C/FRONT, чтобы соединиться с сервером и открыть базу данных, указанными на форме. При этом используется TCP/IP протокол. Код открывает фирму, указанную в поле **MyCompany**. Далее открывается таблица с указанным номером и подсчитывает количество записей в ней. Затем отображается сообщение, в котором указано количество записей в таблице. При нажатии на кнопку OK, будут закрыты таблица и фирма, и произойдет отключение от сервера.

> **Примечание**
>
> Данная форма использует аутентификацию Windows для подключения к серверу. Это указывается в седьмом параметре функции ConnectServerAndOpenDatabase. Если нужно использовать аутентификацию базы данных, то укажите в данном параметре False и передайте логин и пароль пользователя в восьмом и девятом параметре, соответственно.

При завершении, форма должна выглядеть следующим образом:



Обработка Исключений

Как указывалось ранее, нельзя использовать механизм обработки исключений описанный, например, в книге *Inside OLE*. В следующем примере будет показано, как обрабатывать исключения другим способом (но только в компонентах, разработанных вами). Примеры C/OCX устанавливаются вместе с SDK/C/FRONT.

У компонента есть два свойства: Error и ErrorCode, которые используются при обработке исключений. Каждый раз при вызове метода компонента используется свойство *Error* (Boolean), в свойстве устанавливается значение TRUE, если произошла ошибка и FALSE, если ошибки не было. Когда происходит ошибка (Error = TRUE), в ErrorCode устанавливается номер ошибки, который впоследствии может использоваться (например, для отображения соответствующего сообщения для пользователя user).

```
IF (Fin.Error) THEN
  // выполняется обработка ошибки, например:
  ErrorHandlerFunction(Fin.ErrorCode)
ELSE
  // продолжается обработка
```

DRAFT

ЧАСТЬ 7 ДАТАПОРТЫ

ГЛАВА 20 ДАТАПОРТЫ

Датапорты используются для импорта данных во внешние текстовые файлы и экспорта данных из этих файлов.

- Что Такое Датапорт?
- Создание Датапортов
- Экспорт Данных
- Импорт Данных

20.1 Что такое Датапорт?

Датапорты – это объекты, используемые для импорта данных во внешние текстовые файлы и экспорта данных из этих файлов. Существуют широкие возможности выбора при определении формата внешнего файла в процессе импорта и экспорта.

Вы можете контролировать, что происходит при импорте, если запись в импортируемом файле имеет то же значение ключа, что и запись, существующая в таблице базы данных. Кроме того, вы можете контролировать на уровне поля базы данных, нужно ли запускать триггер OnValidate для каждого поля.

Датапорты могут быть динамическими, т.е. при выполнении датапорта определяет, является ли процесс импортом или экспортом, а также имя файла, из которого будут считываться данные или в который они будут записываться. Это достигается посредством опций, которые определяются в форме пользовательского запроса или путем программирования.

На следующей диаграмме представлены компоненты объекта порта данных.

DRAFT

На приведенной диаграмме показано, что датапорт состоит из ряда различных элементов. Далее следует краткое описание каждого элемента.

Описание датапорта

Здесь представлено полное описание датапорта: того, как собираются данные, как они форматируются при записи в выходной файл и т.д. Описание датапорта хранится в базе данных.

Элемент данных

Элемент данных (data item) относится к таблице. Чтобы извлекать информацию из таблиц базы данных, вы определяете элементы данных.

Поля датапорта

Поле датапорта (dataport field) относится к полю в файле, из которого должны импортироваться данные. Поля датапорта во внешнем файле определяются либо как поля с фиксированной длиной, либо как разделенные определенными символами, которые определяете вы.

Форма запроса

Форма запроса (request form) – это форма, которая выполняется до того, как фактически начинает функционировать датапорт. Она используется для получения запросов и опций от пользователя датапорта – например, имени внешнего файла.

Свойство

Свойство (property) – это атрибут объекта (датапорта, элемента данных, поля датапорта и тому подобное), характеризующий некоторым образом данный объект, будь то длина или местоположение поля датапорта в строке (при импорте). Свойства устанавливаются в закладке Property Sheet этого объекта.

Триггер

Предопределенные события, которые происходят с датапортом, позволяют системе выполнять определяемую пользователем C/AL-функцию – событие запускает функцию. Как видно из диаграммы, сам датапорт, элементы данных, поля датапорта, форма запроса и элементы управления, имеющиеся на форме запроса, содержат триггеры. Триггеры редактируются в C/AL Editor.

Логическая структура

Конструирование порта данных включает две основные задачи: конструирование модели данных и определение каркаса внешнего файла.

DRAFT

Модель данных

Определяя элементы данных, вы строите модель данных. Элемент данных относится к таблице.

Экспорт

При экспорте данных каждый элемент данных повторяется для всех записей соответствующей таблицы. Вы можете устанавливать порядок сортировки, ключи и фильтры для таблицы, которые будут использоваться при экспорте. Вы можете также решать, будет ли конкретная запись заноситься во внешний файл.

Импорт

При импорте данных записи, считанные из внешнего файла, могут вставляться в таблицу, относящуюся к данному элементу данных. Вы можете проверить записи, перед тем как вставлять их. Кроме того, можно указать:

- Должны ли записи вставляться в базу данных.
- Должны ли записи вставляться автоматически.
- Должны ли записи, уже находящиеся в БД, быть перезаписаны или обновлены, когда запись с таким же первичным ключом считывается из внешнего файла.

Внешний Файл

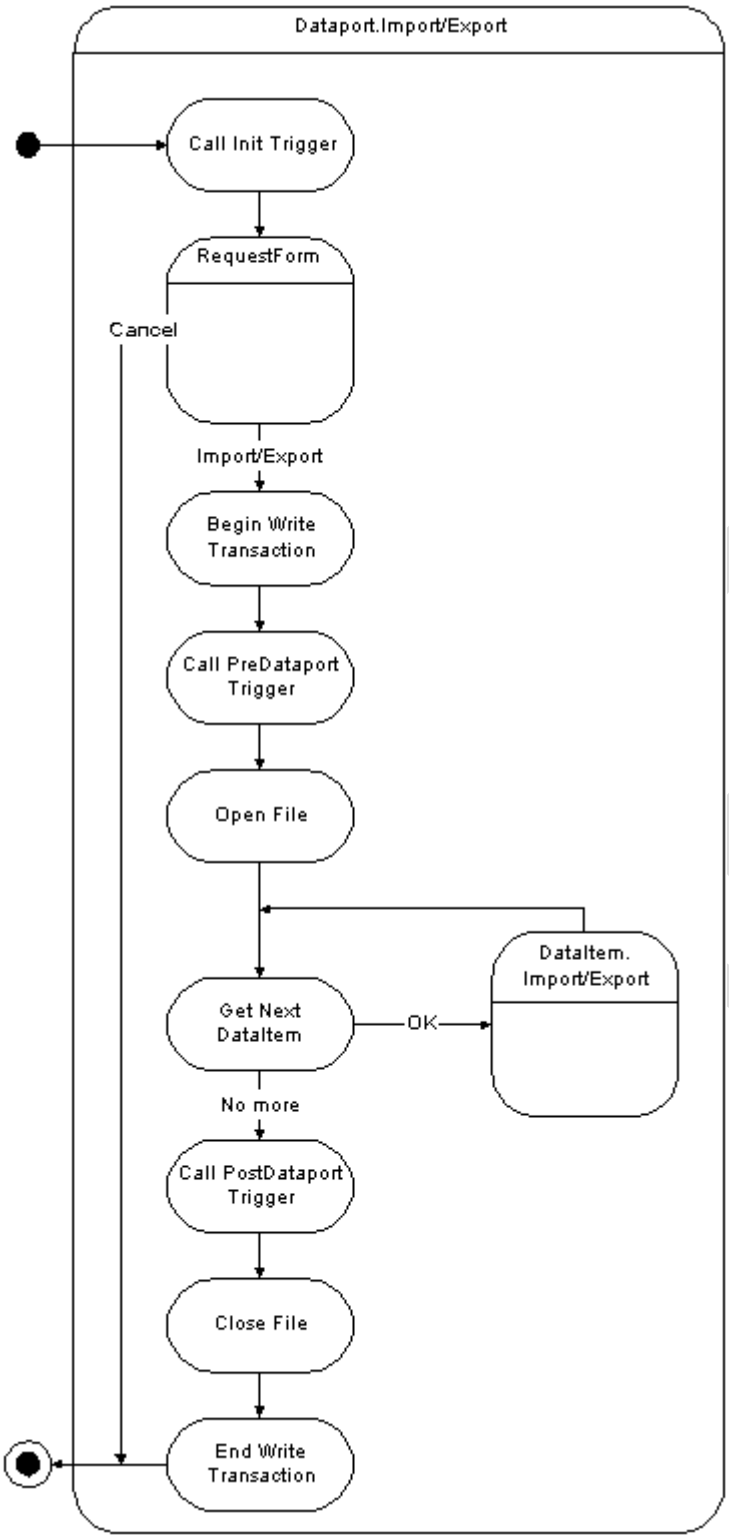
Компоновка внешнего файла определяется свойствами датапорта. Во время импорта эти свойства описывают, как входной поток должен разбиваться на записи и поля датапорта; во время экспорта эти свойства описывают, как поля и записи датапорта должны заноситься в файл.

Свойства, которые можно установить сильно зависят от формата внешнего файла.

Как Работает Датапорт

Следующая диаграмма представляет собой упрощенную версию диаграммы из Приложения C, "Dataport Flow Charts".

DRAFT



1. При запуске датапорта, выполняется триггер OnInitDataport. Данный триггер можно использоваться для инициализации переменных, но не для выполнения операций с данными.

2. После выполнения триггера OnInitDataport запускается форма запроса датапорта (если она была определена). Здесь пользователь может отменить выполнение датапорта.

3. Если пользователь выбирает продолжение выполнения, датапорт начинает транзакцию (запускается команда BWT – Begin Write Transaction – начать транзакцию), после чего вызывается триггер OnPreDataport. На этом этапе данные еще не обрабатываются.

4 Триггер OnPreDataport может использоваться для обработки данных, введенных пользователем на форме запроса.

5. Когда триггер OnPreDataport выполнен, открывается внешний файл, и начинается обработка первого элемента данных.

6. После завершения обработки первого элемента данных следующий элемент данных (если он есть) обрабатывается таким же образом.

7. Если больше нет элементов данных, вызывается триггер OnPostDataport. Вы можете использовать этот триггер для осуществления любой необходимой пост-обработки

8. После завершения работы OnPostDataport внешний файл закрывается.

9 Транзакция, начатая на шаге 3, завершается запуском команды EWT (End Write Transaction – завершить транзакцию)

Обработка каждого элемента данных (шаги 5 и 6) производится, разумеется, при импорте и экспорте по-разному. Подробности показаны на блок-схемах в Приложении С. Наиболее важный принцип, отображенный на общей схеме, заключается в том, что вся обработка датапорта осуществляется во время транзакции. Это означает, что, если обработка прервана в какой-то момент до окончательной команды EWT, никаких следов прерывания выполнения в базе данных впоследствии не будет (однако внешний файл может быть поврежден).

Сохранение, Компиляция и Запуск Датапорта

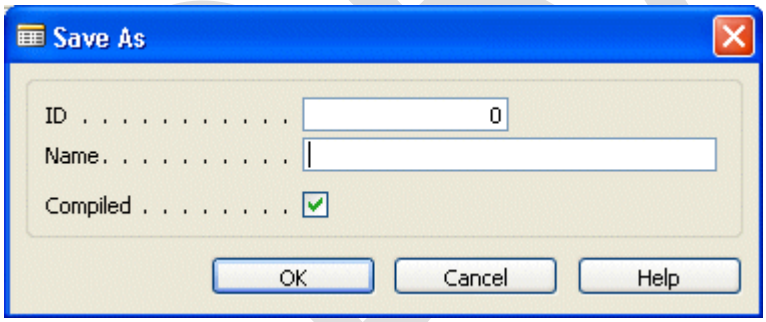
После завершения работы над датапортом его нужно сохранить и откомпилировать, прежде чем его можно будет запускать. Обычно это делается после окончания работ над датапортом. Однако датапорт можно сохранить, даже если работы над ним еще ведутся (из-за чего его нельзя откомпилировать). Кроме того, можно выполнить тестовую компиляцию, не сохраняя и не закрывая датапорт.

Сохранение и Закрытие Датапорта

Датапорт закрывается при закрытии ***Dataport Designer***.

Чтобы сохранить датапорт:

1. Щелкните Файл, Save и укажите имя и идентификатор (ID) датапорта.



Идентификатор должен быть уникальным и соответствовать правилам присваивания номеров объектам. Для дополнительной информации об идентификаторах объектов обратитесь к сертифицированному партнеру Microsoft Business Solutions.

> **Примечание**
>
> Если в свойствах датапорта заполнить свойства ID и Name, то будут использоваться эти значения и система не предложит ввести ID и Name при закрытии датапорта.

2 Значение поля опции **Compiled** по умолчанию - TRUE (отображается как флажок). Если ваш датапорт еще не готов к компиляции, снимите флажок, щелкнув в поле.

3. Щелкните кнопку OK, чтобы сохранить датапорт.

Вы можете сохранить датапорт, не закрывая его, выбрать пункт Save или Save As в меню Файл. Используя Save As, вы можете переименовать существующий датапорт (помните, что вы можете скопировать датапорт, открыв и сохранив его под новым именем).

Компиляция Датапорта

Датапорты, подобно другим объектам в среде C/SIDE, должны компилироваться до того, как их можно будет запускать. Как описано выше, вы можете выбрать возможность компиляции датапорта тогда, когда сохраняете его.

Во время создания датапорта, возможно, потребуется произвести его тестовую компиляцию, чтобы найти возможные ошибки. Для этого нужно щелкнуть Сервис, Компилировать (F11).

Запуск Датапорта

В финальном приложении датапорт будет включен в меню, также он может, например, вызываться командной кнопкой, расположенной на форме. Однако при создании датапорта часто его требуется запускать до того, как он будет интегрирован в приложение.

Тестовое выполнение датапорта

При создании датапорта можно выполнить тестовый запуск, щелкнув Файл, Run (CTRL+R). В результате датапорт будет откомпилирован и запущен на данном этапе разработки. Он не будет сохранен, следовательно, вы можете использовать эту функцию для проверки того, приносят ли вносимые вами изменения ожидаемые результаты, прежде чем сохранять их.

> **Примечание**
>
> Если датапорт используется для импорта, записи не будут сохраняться в таблице базы данных во время тестового выполнения.

Запуск датапорта из Object Designer

Чтобы запустить датапорт из Object Designer, выберите его в списке датапортов и щелкните кнопку Run.

20.2 Создание Датапортов

В данном разделе приводится обзор элементов, которые используются при создании датапорта. В таблицах приведенные описания свойств и триггеров, а также указано для чего используется каждое свойство и триггер. Полные сведения даны в онлайновой справке *C/SIDE Reference Guide*.

Создание датапорта заключается в основном в установке его различных свойств. В последующих разделах даны объяснения того, какие свойства надо использовать и как это делать.

Свойства Датапорта

Данный набор свойств определяет датапорт в целом. Кроме того, здесь вы определяете формат внешнего файла.

Обратите внимание, что свойства Import и FileName могут устанавливаться динамически. Например, вы можете создать датапорт, где пользователь может выбрать возможность импорта и экспорта или выбрать имя внешнего файла для чтения из него данных или записи данных в него (либо вы можете определить, что имя файла будет генерироваться автоматически при запуске датапорта).

Свойство	Назначение
ID	Идентификатор датапорта: должен быть уникален, чтобы его можно было отличить от других датапортов
Name	Имя датапорта: должно быть уникально, чтобы его можно было отличить от других датапортов.

DRAFT

Свойство	Назначение
Caption	Заголовок (отображается, например, в окне формы запроса – по умолчанию такой же, как Name).
CaptionML	Список переводов заголовка объекта.
Import	Это свойство определяет, импортируются или экспортируются данные портом данных. Оно может быть установлено динамически в триггере OnPreDataPort. Оно не может изменяться после запуска триггера OnPreDataPort. Значение по умолчанию <Да> позволяет пользователю выбирать импорт или экспорт на вкладке Параметры формы запроса. Установка значения Да или Нет означает, что пользователь лишен возможности выбора, и задает либо импорт, либо экспорт, соответственно
FileName	Имя внешнего файла, в который записываются или из которого считываются данные. Значение этого свойства может устанавливаться динамически. Если вы переустановите имя файла после того, как файл был открыт, этот файл закрывается, и открывается новый файл
FileFormat	Определяет формат внешнего файла: Variable или Fixed
FieldStartDelimiter	Когда значение свойства FileFormat – Variable, свойство FieldStartDelimiter используется для определения строки, указывающей на начало поля датапорта при вводе или выводе
FieldEndDelimiter	Когда значение свойства FileFormat – Variable, свойство FieldEndDelimiter используется для определения строки, указывающей на конец поля датапорта при вводе или выводе.
FieldSeparator	Когда значение свойства FileFormat – Variable, свойство FieldSeparator используется для определения строки, разделяющей поля датапорта при вводе или выводе.
RecordSeparator	Это свойство используется для определения строки, которая разделяет записи при вводе или выводе
DataItemSeparator	Это свойство используется для определения строки, которая разделяет элементы данных при вводе или выводе
UseReqForm	Это свойство определяет, должна ли форма запроса запускаться до того, как запущен датапорт.
ShowStatus	Это свойство определяет, будет ли выполняться окно состояния во время функционирования порта данных. Кроме того, данное окно содержит кнопку Cancel, которая позволяет прервать выполнение датапорта – что возможно также в случае, если вы создадите собственное диалоговое окно.
TransactionType	Указывается тип транзакции Dynamics NAV. Всего доступно четыре типа базовых транзакций: Browse, Snapshot, UpdateNoLocks and Update. Каждый тип транзакции определяет поведение транзакции Dynamics NAV.
Permissions	Разрешения для датапорта получать доступ к объектам базы данных. Датапорт может иметь больше полномочий, чем отдельный пользователь, что побуждает пользователя использовать датапорты, извлекающие информацию из таблиц, к которым пользователь обычно не может получить доступ.

DRAFT

Свойства *FieldStartDelimiter* и *FieldEndDelimiter* используются для заключения значения поля в кавычки в той ситуации, когда в поле есть символ, определенный как разделитель (FieldSeparator, RecordSeparator или DataItemSeparator). Заполнять данные свойства не обязательно. При импорте, данные ограничители могут использоваться только для того поля, в котором есть символы, указанные как разделители, остальные поля можно оставить так. При экспорте, во внешний файл все поля будут записаны с разделителями и ограничителями.

Формат Файла

Свойство FileFormat определяет формат внешнего файла. Это свойство определяет, как запись считывается из файла или заносится в него. Свойство RecordSeparator определяет, как файл разбивается на записи, после чего свойство FileFormat определяет, как разбить каждую запись на поля датапорта. Наконец, свойство DataItemSeparator определяет, как должны разделяться элементы данных, если датапорт содержит несколько элементов данных. Обратите внимание, что элементы данных не могут быть вложенными, несмотря на то, что датапорт может содержать несколько элементов данных, которые обрабатываются поочередно.

> **Примечание**
>
> При указании свойства FileFormat изменяется доступность некоторых свойств датапорта. Например, если установить FileFormat = Variable, то будут доступны свойства *FieldEndDelimiter*, *FieldStartDelimiter* и *FieldSeparator*.

FileFormat: Fixed

Когда формат внешнего файла – Fixed, поля датапорта в записи имеют фиксированную ширину. Вы можете определить стартовую позицию и ширину каждого поля датапорта в записи (если свойство RecordSeparator – символ newline, запись можно представлять себе как строку текста). Позиции и ширина полей датапорта – это свойства полей датапорта (что описано в одном из следующих подразделов).

FileFormat: Variable

Когда формат внешнего файла – Variable, поля датапорта в записи разделены символами, которые вы определяете, и поля датапорта могут иметь различную ширину. Поля датапорта разделяются строкой (символом), определяемой как свойство FieldSeparator.

Свойства Элемента Данных

Этот набор свойств описывает элементы данных датапорта. Элемент данных связан с таблицей базы данных.

Большая часть этих свойств та же и имеет те же функции, что и соответствующие свойства элемента данных в отчете ГЛАВА 12 ОТЧЕТЫ – БАЗОВЫЕ ПОНЯТИЯ. Три свойства являются особенностью датапорта: AutoSave, AutoUpdate и AutoReplace.

Свойство	Назначение
DataItemIndent	Уровень вложенности (можно задать из дизайнера, при создании элемента данных)
DataItemTable	Таблица элемента (может устанавливаться в дизайнере при создании элементов данных).
DataItemVarName	Имя записи как переменной (по умолчанию – имя DataItemTable).
DataItemTableView	Применяемый ключ, порядок сортировки и фильтры
ReqFilterHeading	Заголовок закладки для данного элемента на форме запроса (по умолчанию – имя DataItemTable).

DRAFT

Свойство	Назначение
ReqFilterHeadingML	Перевод ReqFilterHeading.
ReqFilterFields	Имена полей, которые будут первоначально включены в форму ReqFilter.
CalcFields	Имена полей, которые будут вычисляться после извлечения записи.
DataItemLinkReference	DataItemVarName элемента данных с меньшим уровнем вложенности. Для подчинения текущего элемента данных другому элементу данных.
DataItemLink	Связь между полями текущего элемента данных и элемента данных указанного в свойстве DataItemLinkReference.
AutoSave	Это свойство определяет, будут ли импортированные записи автоматически вставляться в таблицу C/SIDE.
AutoUpdate	Это свойство определяет, будут ли импортированные записи инициализированы значениями из существующей записи с тем же первичным ключом.
AutoReplace	Это свойство определяет, будут ли импортированные записи автоматически заменять существующие записи с тем же первичным ключом.

AutoUpdate, AutoReplace, AutoSave

Эти три свойства с приставкой Auto* определяют, как обрабатываются записи, считанные из внешнего файла. Кроме того, они используются для разрешения конфликта, возникающего в том случае, когда запись, считанная из внешнего файла при импорте, имеет такой же первичный ключ, как и запись, уже существующая в таблице базы данных.

Свойства AutoSave и AutoReplace используются для определения того, как записи сохраняются таблице базы данных. Обратите внимание, что если значение свойства AutoSave - Нет, то установки AutoReplace и AutoUpdate не действуют – в этом случае вы должны обрабатывать конфликты, используя C/AL-код.

В следующей таблице приводится список результатов, к которым приводят различные комбинации значений этих свойств:

AutoSave	Auto Update	Auto Replace	Запись есть в базе данных и импортируемом файле	Запись есть только в импортируемом файле
Нет*	---	---	Запись в базе данные не обновляется и не заменяется автоматически	Импортируемая запись не вставляется в базу данных автоматически
Да	Нет	Нет	Произойдет ошибка выполнения, и импорт будет завершен	Импортируемая запись вставляется в базе данных автоматически
Да	Нет	Да	Импортируемая запись заменит существующую.	Импортируемая запись вставляется в базу данных автоматически
Да	Да	---	Импортируемая запись обновит существующую запись в базе данных	Импортируемая запись вставляется в базу данных автоматически

DRAFT

--- настройка не имеет значения.

* Если AutoSave – Нет, вставлять или модифицировать запись, можно используя функции вставки или модификации из C/AL.

Функция AutoUpdate полезна в некоторых конкретных ситуациях. Ее действие лучше всего объяснить на небольшом примере.

Пример

Предположим, у вас имеется таблица, являющаяся списком товаров. Вы обновляете цены, экспортируя список с номерами товаров (первичный ключ) и ценами во внешний файл, после чего вычисляете цены в электронной таблице. Теперь, когда цены определены, вы готовы импортировать файл с новыми ценами. Очевидно, что записи, считанные из внешнего файла, будут иметь тот же первичный ключ, что и записи, которые уже находятся в базе данных. Использование свойств AutoSave и AutoReplace не позволит решить проблему. Если вы будете заменять каждую запись соответствующей записью из импортируемого файла, вся информация, за исключением номеров товаров и цен, будет потеряна (предполагается, что в таблице содержится не только информация о номерах товаров и ценах, но и, например, названия товаров, уровень запасов и т.д.).

Решение заключается в использовании свойства AutoUpdate. При импорте запись фактически заменяет существующую запись – но поля, отсутствующие в импортируемой записи, инициализируются данными из уже существующей записи, а не оставляются пустыми. Существующая запись обновляется с использованием измененных данных.

Свойства Поля Датапорта

Этот набор свойств определяет поля записи. Когда значением свойства FileFormat = Fixed, вы используете свойства StartPos и Width каждого поля датапорта для определения того, как запись, которая считывается из внешнего файла, разбивается на поля датапорта. При экспорте эти свойства определяют, как данные записываются из базы данных во внешний файл.

Свойство	Назначение
Enabled	Определяет, должно ли поле датапорта быть включено или отключено. Отключенное поле датапорта импортируется из внешнего файла, но не будет вставлено в запись.
SourceExpr	Исходное выражение поля порта данных. При импорте им может быть имя поля таблицы, где должно сохраняться считываемое из внешнего файла значение, но им может быть любая допустимая переменная C/AL. При экспорте им может быть значение, экспортируемое во внешний файл, - например, имя поля таблицы, но им может быть любое допустимое выражение C/AL.
Caption	Заголовок на текущем языке. Значение берется из свойства *CaptionML,* если оно заполнено. Значение по умолчанию совпадает со значением свойства name поля
CaptionML	Список переводов свойства caption.
StartPos	Если значением свойства FileFormat является Fixed, это – стартовая позиция данного поля порта данных. Нумерация позиций начинается с 1.
Width	Если значением свойства FileFormat является Fixed, это – ширина данного поля датапорта.
CallFieldValidate	Определяет, будет ли выполняться триггер OnValidate при импорте поля датапорта

DRAFT

Свойство	Назначение
Format	Описывает, как поле датапорта будет форматироваться во время экспорта. Например, вы можете определить количество десятичных разрядов и т.д.
AutoCalcField	Система автоматически рассчитает значения вычисляемого поля (FlowField)

Перед выполнением экспорта поля, данные преобразуются в текст. Если формат имеет значение Fixed, а ширина (Width) меньше, чем фактическая ширина строки данных после преобразования, содержимое будет усечено справа так, чтобы оно имело определенную ширину. То есть число не округляется, а усекается как текст – справа.

Вы получите предупреждение во время конструирования, если вы определили поля датапорта с такими стартовыми позициями и шириной, которые приведут к перекрытию этих полей датапорта. Ошибка делает невозможной компиляцию (и, соответственно, выполнение) датапорта.

Однако вы можете иметь разрывы в полях датапорта (стартовая позиция второго поля не обязательно должна быть там, где закончилось первое поле). Благодаря этим разрывам можно пропускать ненужную информацию или создавать пустые столбцы в файле.

Во время импорта значения, которое слишком велико для конкретного типа данных или заданной ширины поля таблицы базы данных, куда оно должно быть вставлено, вызовет ошибку, и выполнение прекратится. Поскольку это произойдет в транзакции, аварийное завершение не испортит базу данных.

Триггеры Датапорта

В следующей таблице перечислены триггеры запускаемые, при работе датапорта. Здесь приведено лишь краткое описание. Более полное описание содержится в онлайновой справке *C/SIDE Reference Guide*.

Триггер	Выполняется
OnInitDataport	Когда датапорт загружен, но перед тем как будет запущена форма запроса и установлены фильтры и ключи для таблиц.
OnPreDataport	Перед запуском датапорта, но после того как была запущена форма запроса. Ключи и фильтры уже установлены, когда запускается данный триггер
OnPostDataport	Когда все элементы данных уже обработаны
OnPreDataItem	Перед тем как элемент данных начнет обрабатываться, но все соответствующие переменные уже инициализированы, а ключи и фильтры таблицы уже установлены
OnBeforeExportRecord	Когда запись извлечена и готова для экспорта.
OnAfterExportRecord	После того как запись была экспортирована во внешний файл. Данный триггер можно использовать, чтобы выполнить обработку внешнего файла, перед тем как загружать очередную запись, например, передвинуть указатель.

DRAFT

Триггер	Выполняется
OnBeforeImportRecord	Перед тем как запись будет импортирована из внешнего файла. Можно использовать данный триггер, чтобы выполнить обработку внешнего файла, перед тем как загружать очередную запись, например, передвинуть указатель.
OnAfterImportRecord	После того как запись была импортирована из внешнего файла, но перед тем как она будет вставлена в таблицу. Этот триггер можно использовать, чтобы выполнить обработку записи, перед тем как она будет вставлена, например, отказаться от вставки этой записи.
OnPostDataItem	После того как была обработана последняя запись в элементе данных.
OnAfterFormatField	После того как значение поля было отформатировано (преобразовано в текст), но перед тем как текст был записан во внешний файл. Этот триггер предоставляет доступ к отформатированному значению в текстовом формате.
OnBeforeEvaluateField	После того как поле было прочитано из внешнего файла, но перед тем как оно будет преобразовано и проверено. Этот триггер предоставляет доступ к импортированному значению в текстовом формате

20.3 Экспорт Данных

В данном разделе будет описано, как создавать датапорты всех четырех основных типов: на импорт и экспорт, с фиксированным и переменным форматом внешнего файла. В последнем примере будет показано, как создавать датапорт, который выполняет как экспорт, так и импорт, а также обновляет данные при импорте.

Экспорт – Фиксированный Формат

Данный датапорт выполняет экспорт в файл с фиксированным форматом. Каждая запись начинается с новой строки, длина поля фиксирована и не зависит от фактических данных.

В данном примере будет использована таблица ***G/L Account*** (Фин. Счета). Она включает несколько вычисляемых полей, которые также будут экспортированы, для этого придется выполнить их расчет.

Простая Версия

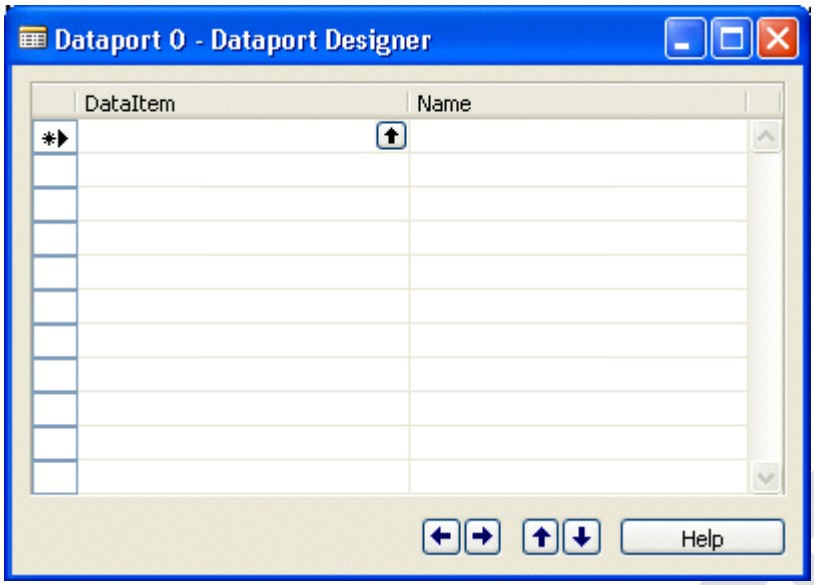
Первая версия будет очень простой, но затем она будет улучшена.

Чтобы создать датапорт:

1. Щелкните Сервис, Object Designer (SHIFT+F12).

2 В **Object Designer**, щелкните кнопку **Dataport**, затем кнопку **New**. Откроется окно ***Dataport Designer***:

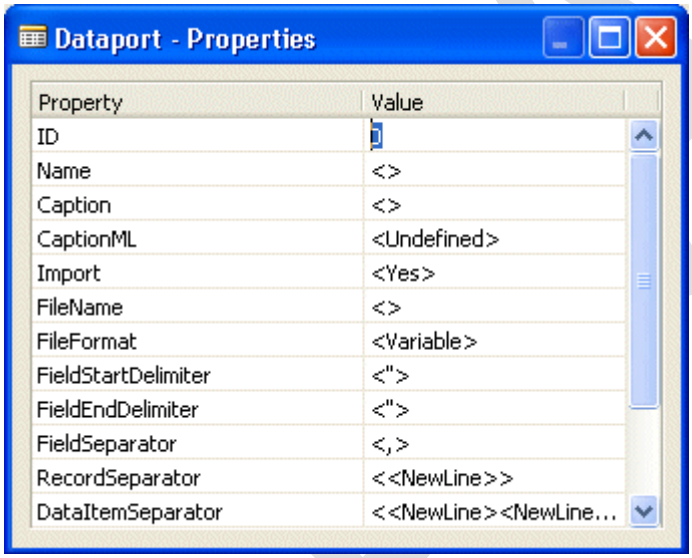
DRAFT



3. В первом поле **DataItem** щелкните кнопку AssistButton и выберите таблицу ***G/L Account*** из окна ***Table List***.

В поле name будет установлено название таблицы. В данном датапорте оставим это поле без изменений.

4. Щелкните по пустой строке в **Dataport Designer**, чтобы выбрать сам датапорт, затем щелкните Вид, Properties (SHIFT+F4), чтобы открыть окно ***Properties*** датапорта (не элемента данных). Также датапорт можно выбрать, щелкнув Правка, Выбрать Объект.



5. В окно ***Properties*** оставьте значения по умолчанию по всех свойствах, кроме:

Import – установите значение Нет, чтобы создать датапорт, экспортирующий данные.

FileFormat – установите значение Fixed.

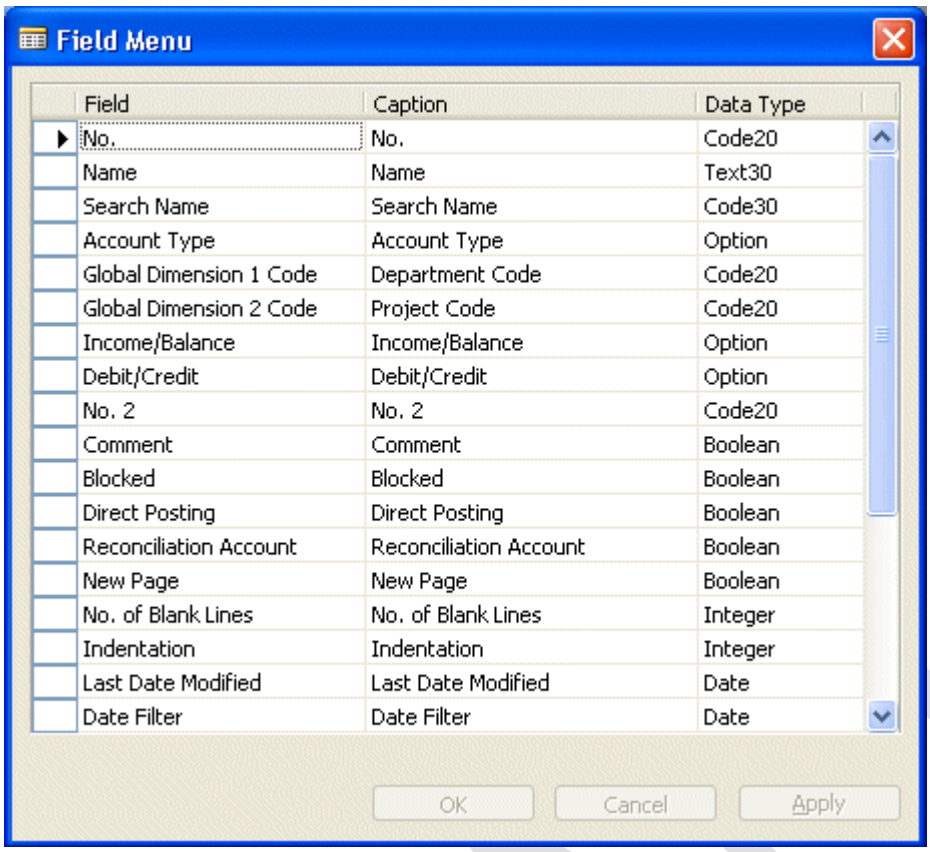
После того как был создан датапорт с одним элементом данных, нужно указать экспортируемые поля.

Добавление полей датапорта

6. В Dataport Designer выберите элемент данных G/L Account.

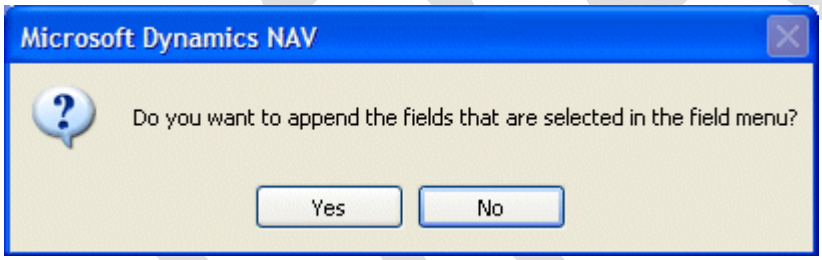
7. Щелкните Вид, Dataport Fields, откроется окно ***Field Designer***.

8. После того как будет открыто окно **Field Designer** щелкните Вид, Field Menu.

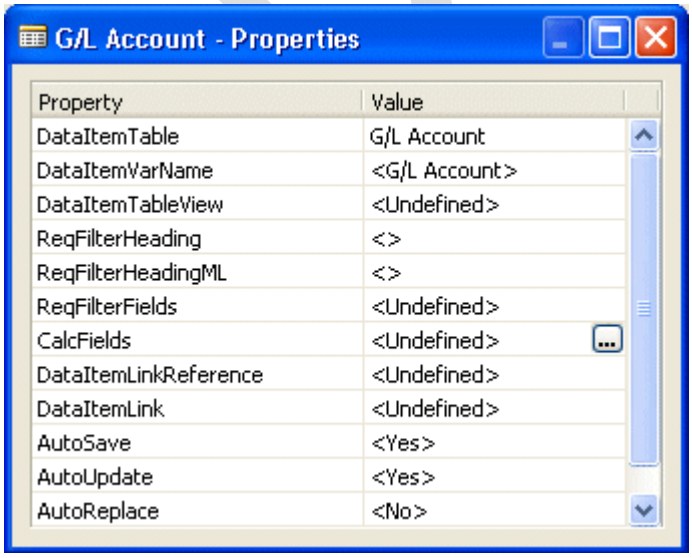


9. В окне ***Field Menu*** выберите поля, которые должны экспортироваться. В данном примере это будут поля: **No.**, **Name**, **Balance at Date** и **Net Change**.

10. Щелкните в **Field Designer**. Система спросит, хотите ли вы добавить выбранные поля. Щелкните кнопку Да.

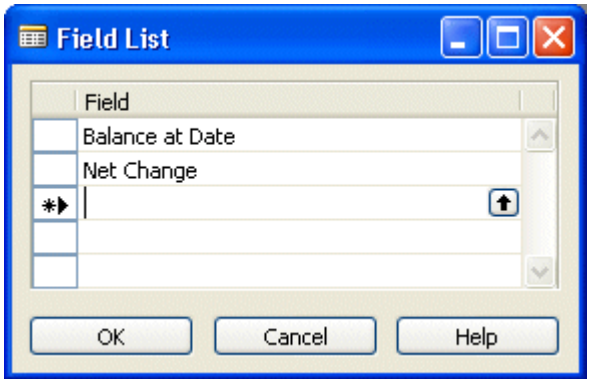


11. Выберите элемент данных G/L Account и откройте для него окно ***Properties*** (SHIFT+F4).



12. Щелкните кнопку AssistButton в свойстве *CalcFields*, чтобы открыть окно ***Field List***.

13. В окне ***Field List*** выберите вычисляемые поля, значения которых должны рассчитываться перед экспортом. В данном примере это поля **Balance at Date** и **Net Change**. Выберите их и щелкните кнопку OK.

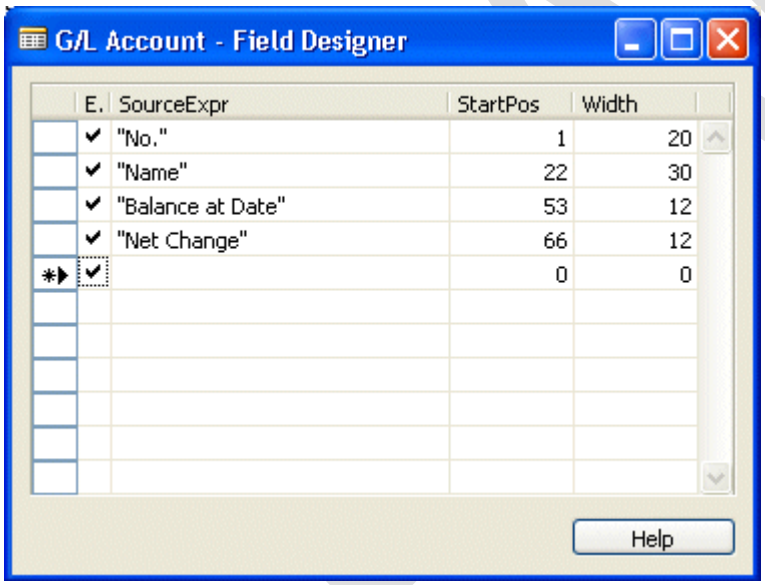


Теперь можно запустить датапорт. Щелкните Файл, Run, чтобы запустить датапорт не сохраняя его. Если вы хотите вначале сохранить датапорт, то потребуется присвоить ему идентификатор и наименование.

Свойства StartPos и Width

После того как с помощью Field Menu были вставлены поля, можно проверить значения *StartPos* и *Width* для всех полей.

Щелкните Вид, Dataport Fields, чтобы открыть окно ***Field Designer***:



Как можно видеть и StartPos, и Width уже заполнены. В них указываются значения на основании следующих правил:

Тип данных поля	Значение в Width
Code	Фактическая длина, если она больше 10, иначе 10
Text	Фактическая длина, если она больше 30, иначе 30
Date	11
Time	10
Option	10
Decimal	12

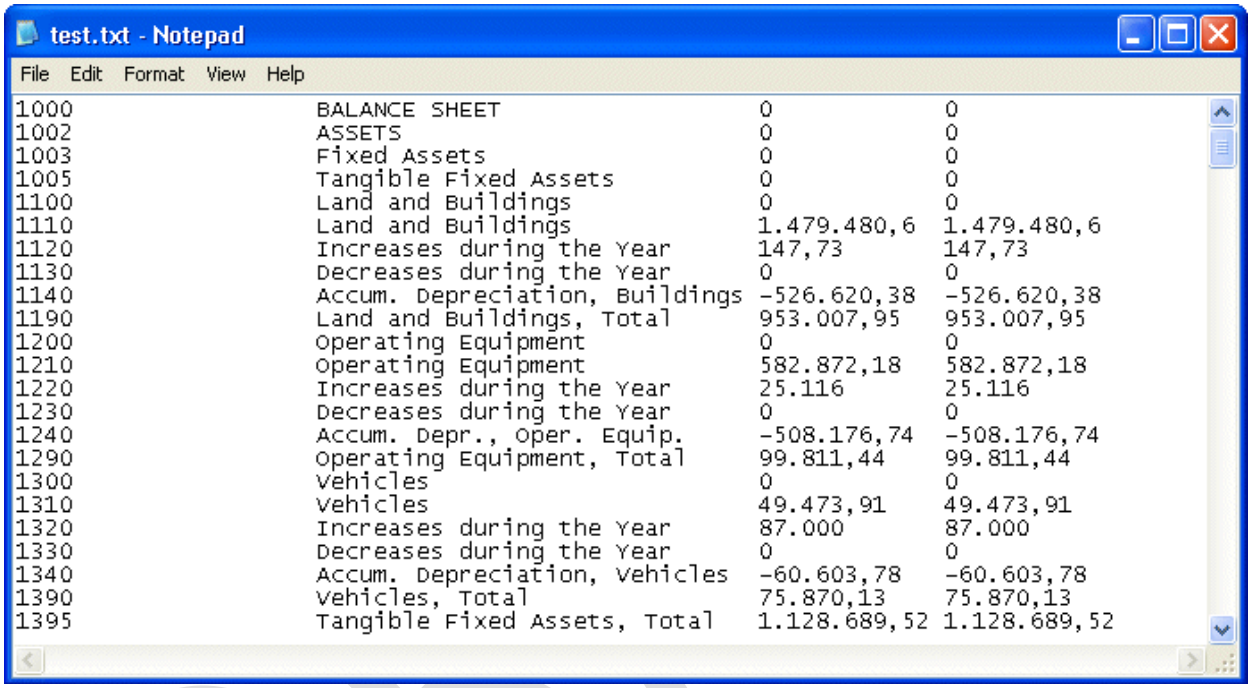
Тип данных поля	Значение в Width
Integer	7
Boolean	10

Запуск датапорта

Когда вы запустили датапорт, появилась форма запроса по умолчанию. Первая закладка малоинтересна, а вот закладка **Параметры** важна, т.к. в ней вводится название внешнего файла, в который будет производиться запись.

Введите название файла и щелкните кнопку OK, чтобы запустить датапорт.

После выполнения датапорта, можно открыть файл текстовым редактором:



1000	BALANCE SHEET	0	0
1002	ASSETS	0	0
1003	Fixed Assets	0	0
1005	Tangible Fixed Assets	0	0
1100	Land and Buildings	0	0
1110	Land and Buildings	1.479.480,6	1.479.480,6
1120	Increases during the Year	147,73	147,73
1130	Decreases during the Year	0	0
1140	Accum. Depreciation, Buildings	-526.620,38	-526.620,38
1190	Land and Buildings, Total	953.007,95	953.007,95
1200	Operating Equipment	0	0
1210	Operating Equipment	582.872,18	582.872,18
1220	Increases during the Year	25.116	25.116
1230	Decreases during the Year	0	0
1240	Accum. Depr., Oper. Equip.	-508.176,74	-508.176,74
1290	Operating Equipment, Total	99.811,44	99.811,44
1300	Vehicles	0	0
1310	Vehicles	49.473,91	49.473,91
1320	Increases during the Year	87.000	87.000
1330	Decreases during the Year	0	0
1340	Accum. Depreciation, Vehicles	-60.603,78	-60.603,78
1390	Vehicles, Total	75.870,13	75.870,13
1395	Tangible Fixed Assets, Total	1.128.689,52	1.128.689,52

Улучшенная Версия

Простой датапорт можно сделать более дружественным пользователю. В зависимости от назначения датапорта, можно улучшить некоторые моменты. Например:

- Первая закладка не должна отображаться, т.к. от пользователя не требуется ни установка фильтров, ни изменение ключей.
- Должны выгружаться только записи, у которых в поле Account Type установлено значение Posting или End-Total.
- Числа должны выгружаться как тысячи. Не должно быть разделителей для тысяч, дробных значений, знак должен находиться в начале поля.

Изменение формы запроса

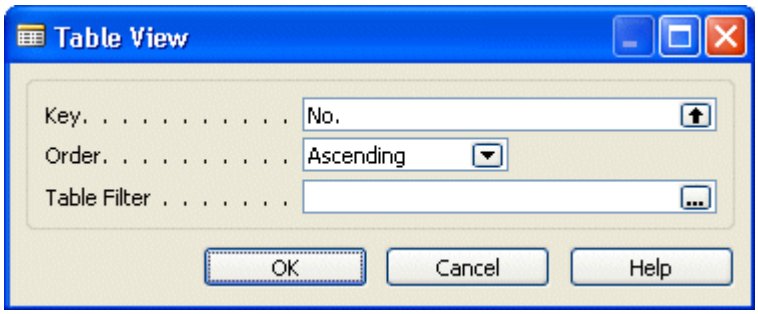
Никогда не устанавливайте в свойстве *UseReqForm* значение Нет, чтобы скрыть форму запроса. В этом случае пользователь не сможет ввести название файла, в который должны записываться данные. Это приведет к ошибке выполнения датапорта потому, что название файла должно быть указано.

Вместо этого можно оставить закладку **Параметры** и убрать закладку, связанную с элементом данных. Для этого нужно использовать свойство *DataItemTableView* элемента данных. Если данное поле заполнено, то пользователь не сможет изменить ключ и установить фильтр, также закладка не будет отображаться.

Чтобы установить значение в свойстве *DataItemTableView* элемента данных:

1. В **Dataport Designer** выберите элемента данных G/L Account и откройте окно ***Properties*** (SHIFT+F4).

2. Щелкните кнопку AssistButton в свойстве *DataItemTableView*, чтобы открыть окно ***Table View***:

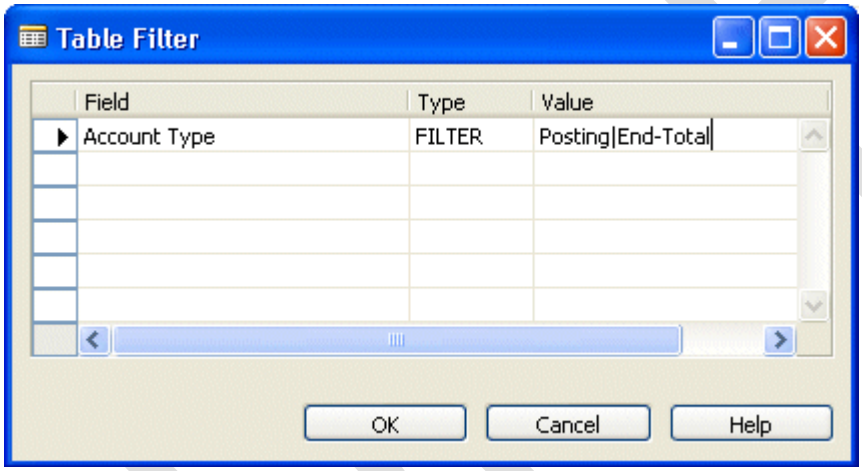


3. Заполните как минимум одно из доступных полей. В данном примере в поле **Key** было установлено значение No., а в поле **Order** - Ascending.

Выбор определенных типов финансовых счетов

Чтобы выбрать определенные типы финансовых счетов нужно установить фильтр таблицы. Это можно сделать с помощью свойства *DataItemTableView*.

4. В поле **Table Filter** щелкните кнопку AssistButton, чтобы открыть окно ***Table Filter***:



5. Установить в полях **Field**, **Type** и **Value** значения согласно рисунку. Таким образом, будет создан фильтр таблицы, при котором будут выбираться только те записи, у которых в поле Account Type установлено либо значение Posting, либо End-Total (символ «|» между двумя значениями означает ИЛИ).

6. Щелкните кнопку OK, чтобы закрыть окна ***Table Filter*** и ***Table View***.

Изменение форматирования чисел

Чтобы выгружать поля **Balance at Date Net Change,** тип, которых Decimal, в виде тысяч (число 1.444.723,67 должно экспортироваться как 1444), нужно использовать свойство *SourceExpr* данных полей:

Чтобы изменить форматирование чисел в полях:

1. Выберите элемент данных G/L Account и щелкните Вид, Dataport Fields, чтобы открыть окно Field Designer.

2 В окне ***Field Designer*** выберите поле **Balance at Date** и щелкните Вид, Properties (SHIFT+F4).

3. Введите следующее выражение в свойство *SourceExpr* данного поля:

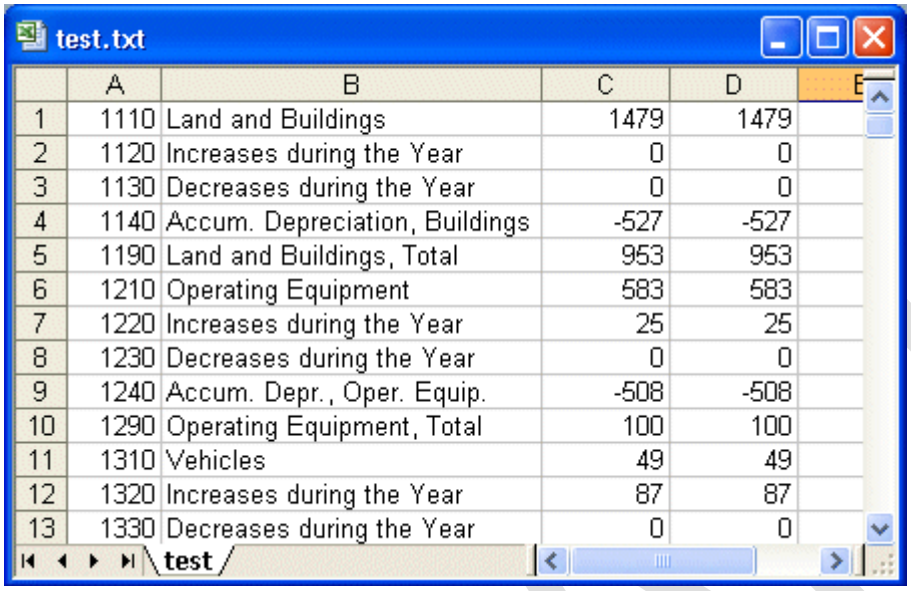
FORMAT(ROUND("Balance at Date"/1000,1,'='),0,1)

В результате вычисления данного выражения на языке C/AL, значение поля **Balance at Date** будет разделено на 1000. Результат деления будет округлен с помощью функции ROUND, а затем, с помощью функции FORMAT будет применен формат по умолчанию 1 (что означает <Sign><Integer><Decimals>).

4. В свойство SourceExpr поля **Net Change** введите следующее выражение:

FORMAT(ROUND("Net Change"/1000,1,'='),0,1)

При открытии, созданного датапортом файла с помощью Microsoft Excel, он будет выглядеть следующим образом:



test.txt

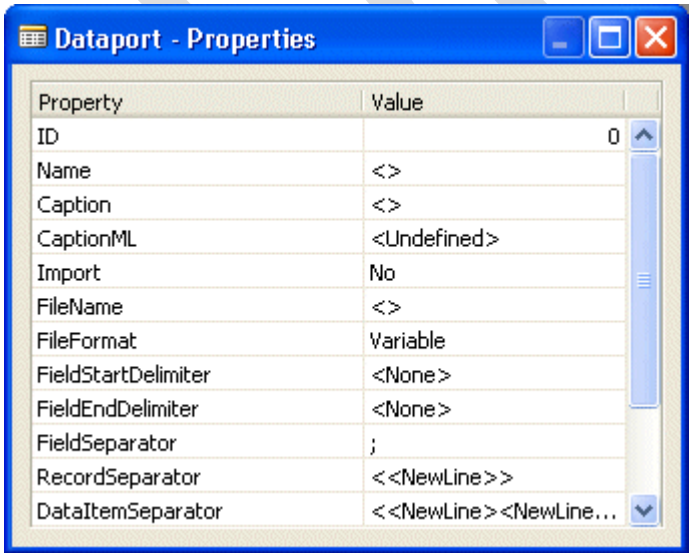
	A	B	C	D
1	1110	Land and Buildings	1479	1479
2	1120	Increases during the Year	0	0
3	1130	Decreases during the Year	0	0
4	1140	Accum. Depreciation, Buildings	-527	-527
5	1190	Land and Buildings, Total	953	953
6	1210	Operating Equipment	583	583
7	1220	Increases during the Year	25	25
8	1230	Decreases during the Year	0	0
9	1240	Accum. Depr., Oper. Equip.	-508	-508
10	1290	Operating Equipment, Total	100	100
11	1310	Vehicles	49	49
12	1320	Increases during the Year	87	87
13	1330	Decreases during the Year	0	0

test

Экспорт – Переменный Формат

Это пример датапорта, который будет выгружать те же записи, что и в предыдущем примере, но в переменном формате. Каждое поле записи будет ограничено указанными символами, и иметь длину, совпадающую с фактической длиной значения в поле.

Чтобы из датапорта с фиксированным форматом получить датапорт с переменным форматом, нужно выполнить несколько изменений. Для начала нужно указать правильный набор свойств для элемента данных, такой как на рисунке ниже:



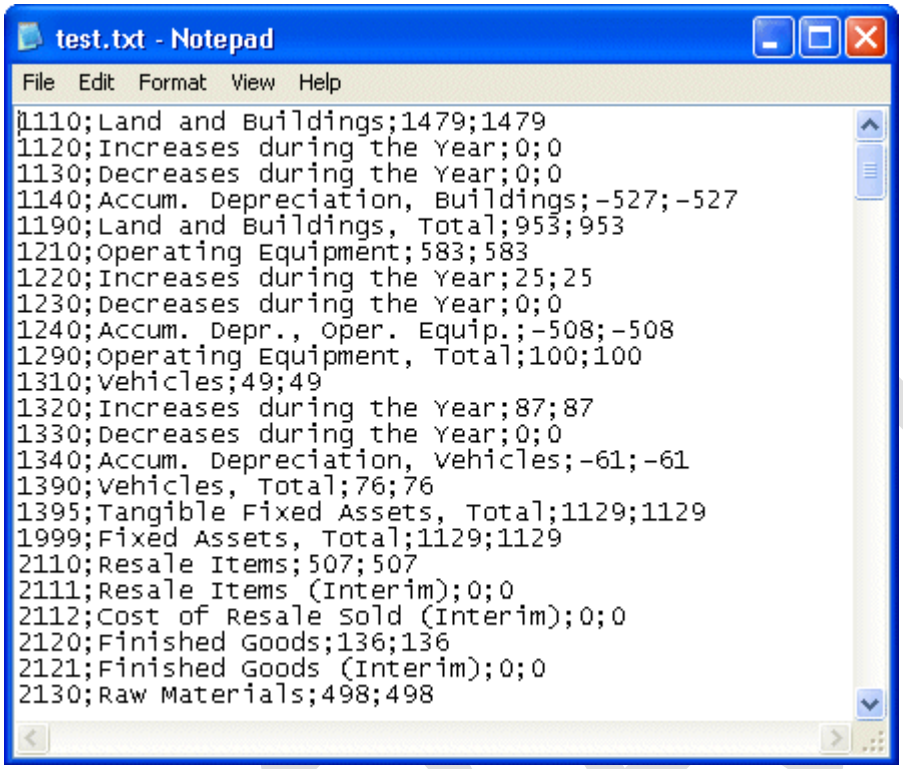
Dataport - Properties

Property	Value
ID	0
Name	<>
Caption	<>
CaptionML	<Undefined>
Import	No
FileName	<>
FileFormat	Variable
FieldStartDelimiter	<None>
FieldEndDelimiter	<None>
FieldSeparator	;
RecordSeparator	<<NewLine>>
DataItemSeparator	<<NewLine><NewLine...

- В свойстве *FileFormat* установить значение Variable.
- В свойстве *FieldSeparator* установить значение «;» (точка с запятой).

- В свойствах *FieldStartDelimiter* и *FieldEndDelimiter* установить значения <None>.
- В свойствах *RecordSeparator* и *DataItemSeparator* оставить значения по умолчанию. Они означают, что каждая новая запись будет выведена в отдельной строке, а следующий элемент данных будет отделен двумя новыми строками (в данном примере это не важно, так как в этом датапорте один элемент данных)

В текстовом редакторе, экспортированный файл будет выглядеть следующим образом:



В качестве FieldSeparator была использована точка с запятой по той причине, что в полях используются и пробелы, и запятые. Другим вариантом могло быть использование ограничителей (delimiters). В этом случае в свойстве FieldSeparator могла быть указана запятая. В любом случае выбор зависит от назначения экспортированного файла и форматов, поддерживаемых приложением, которое будет загружать текстовый файл.

20.4 Импорт Данных

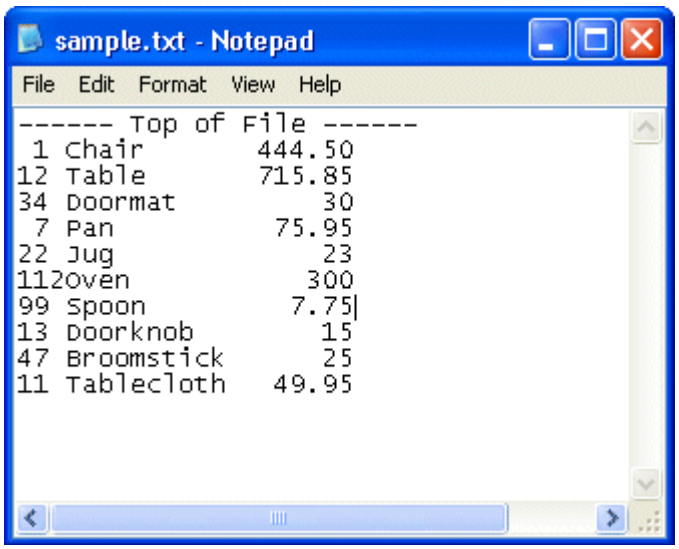
В данном разделе будет описано, как создавать датапорты, импортирующие данные, и использующие два возможных формата внешних файлов: фиксированный и переменный.

Импорт – Фиксированный Формат

Создание датапорта для импортирования данных не сильно отличается от создания датапорта экспортирующего данные. Однако нужно решить, как будут загруженные записи вставляться в таблицу, которая связана с элементом данных. Это особенно актуально, если таблица уже содержит данные, у которых значения первичного ключа совпадает с записями, содержащимися во внешнем файле.

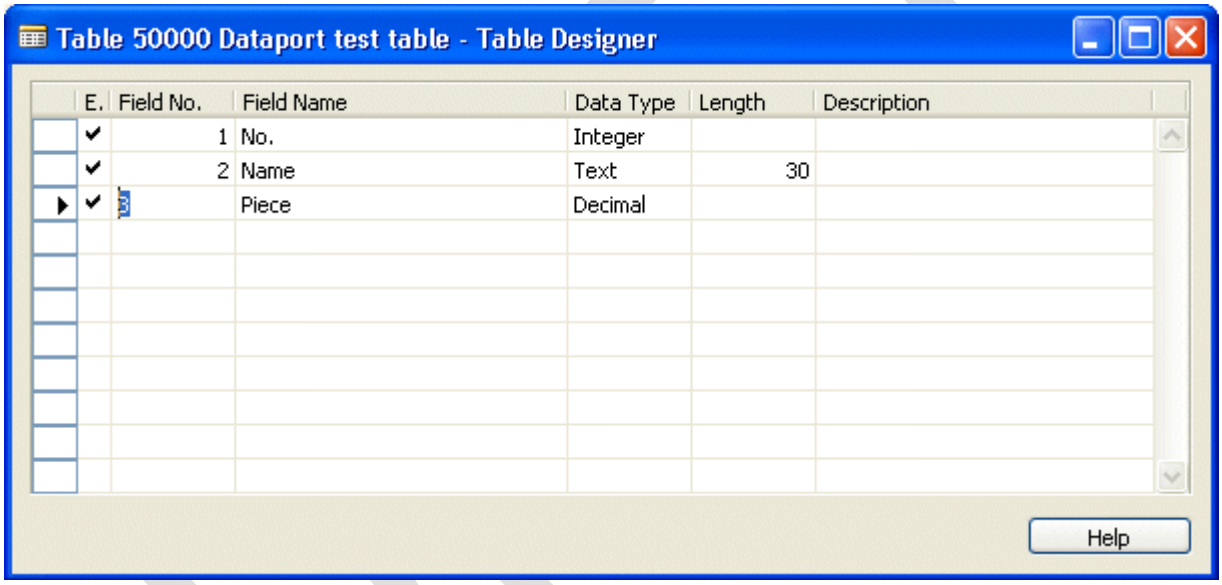
При выполнении первого примера предположим, что таблица пуста (или предприняты другие действия, позволяющие исключить возможность возникновения конфликта). В следующем примере будет показано, что делать, если возможно возникновение конфликтов.

Файл, который вы собираетесь импортировать, выглядит следующим образом:

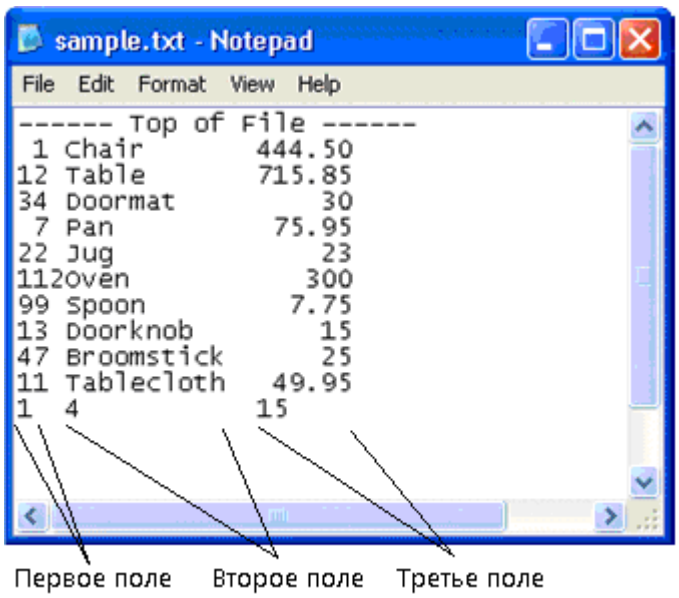


Настройка таблицы

В данном примере записи будут импортированы в новую таблицу. Так выглядит новая таблица в дизайнере:



Это значит, что вы должны определить, как импортируемые строки файла будут становиться записями элемента данных и разбиваться на поля. Строки имеют фиксированный формат, при внимательном рассмотрении строк можно определить начальные позиции полей:



Это означает, что для полей датапорта нужно установить следующие свойства:

Field	StartPos	Width
No.	1	3
Name	4	11
Price	15	6

После того как вы выяснили как будет организован импорт файла, создание датапорта не вызовет проблем:

1. Открыть **Object Designer** и щелкнуть кнопку **Dataport**, затем кнопку **New**, чтобы создать датапорт.

2. Открыть окно ***Properties*** для датапорта и установить свойства *Import* = Да, *FileFormat* = Fixed.

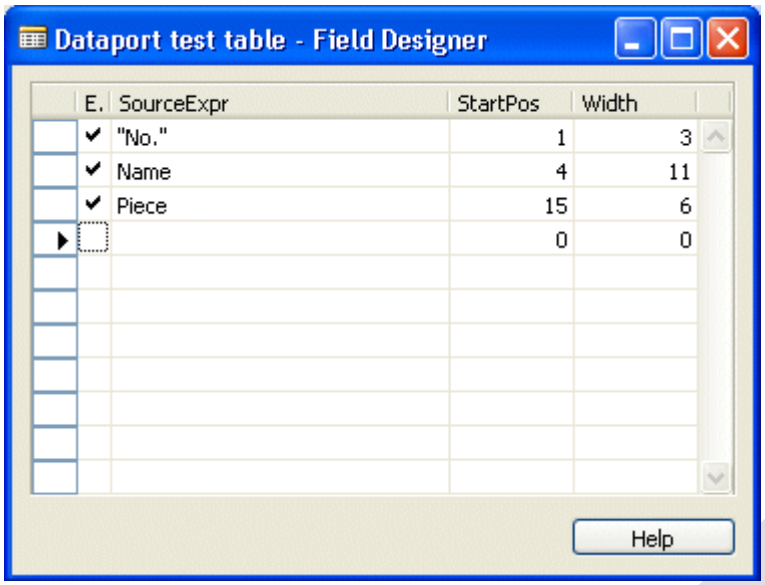
3. В данном примере нужно создать элемент данных на основании созданной ранее таблицы.

4. Щелкнуть Вид, Dataport Fields, чтобы открыть **Field Designer** и добавить все поля из таблицы в датапорт.

5 Нужно изменить значения свойств *StartPos* и *Width* согласно фактическим значениям файла, из которого будет выполняться импорт данных.

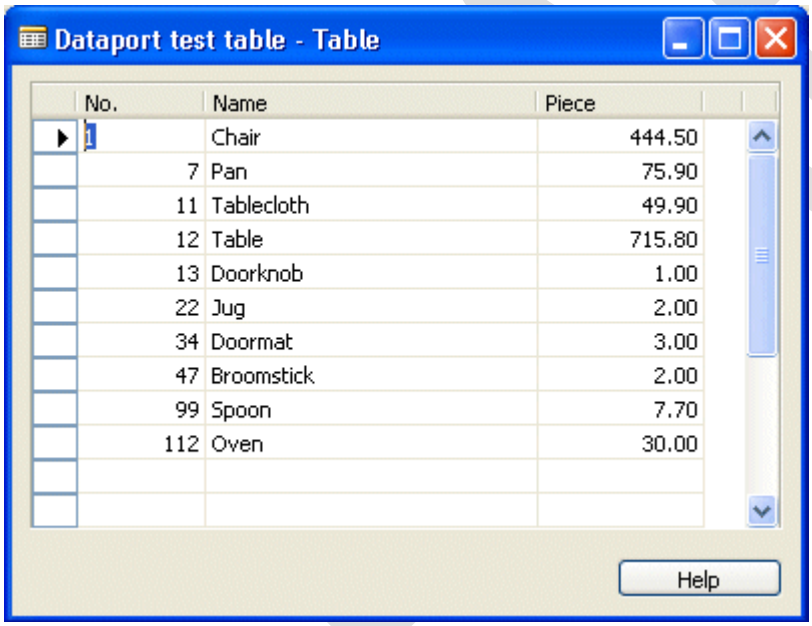
В данном примере свойства должны быть установлены так:

DRAFT



6. Теперь можно запустит датапорт. Помните, что если запустить датапорт из **Dataport Designer** (щелкнув Файл, Run), записи не будет сохранены в базе данных (чтобы сохранить их нужно вызвать датапорт из **Object Designer**, или из меню). Также может потребоваться убрать лишние закладки из формы запроса.

После запуска датапорта со вставкой записей в таблицу, таблица будет выглядеть следующим образом:



Обратите внимание, что поле **Field No.** является первичным ключом таблицы, и порядок, в котором отображаются записи, также определяется первичным ключом, и он отличается от порядка записей в импортируемом файле.

Возможные ошибки

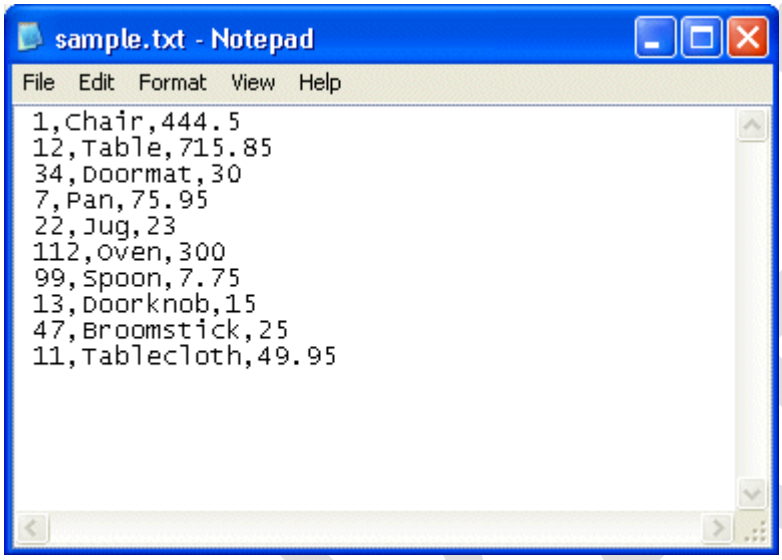
Очень легко сделать ошибку, при определении где «разрезать» строки импортируемого файла. В некоторых случаях это может привести к ошибке выполнения. Например, если вы сделаете ошибку в настройке длины поля **Field No.** и укажете ее на один символ больше. Для большинства строк импортируемого файлов ничего не произойдет, так как система проигнорирует пробелы. Но строка, которая начинается с "112Oven..." вызовет ошибку выполнения, так как среда C/SIDE прочитает 112O вместо 112. Символ "O" (буква «о» в верхнем регистре) не может быть вставлен в поле типа integer.

Можно считать удачей, что такая ошибка вызывает ошибку выполнения. В других случаях среда C/SIDE может не обнаружить ошибку, например, если некорректно разделены два текстовых поля. При возможности тщательно тестируйте ваши датапорты, перед тем как передавать их в промышленную эксплуатацию.

> **Совет**
>
> Если порядок полей в импортируемом файле отличается от порядка в таблице, то в **Field Designer** укажите порядок полей согласно таблице, а значения свойств StartPos и Width рассчитайте вручную.

Импорт – Переменный Формат

Чтобы из датапорта с фиксированным форматом получить датапорт с переменным форматом, нужно выполнить несколько изменений. Предположим, что файл выглядит следующим образом:



Данные в этом файле, такие же, как и в файле с фиксированным форматом, который импортировался ранее, но в данном случае поля разделены запятыми.

Чтобы изменить датапорт так, чтобы он смог импортировать внешний файл с переменным форматом нужно выполнить некоторые изменения в окне ***Properties*** датапорта.

Нужно выполнить следующие изменения:

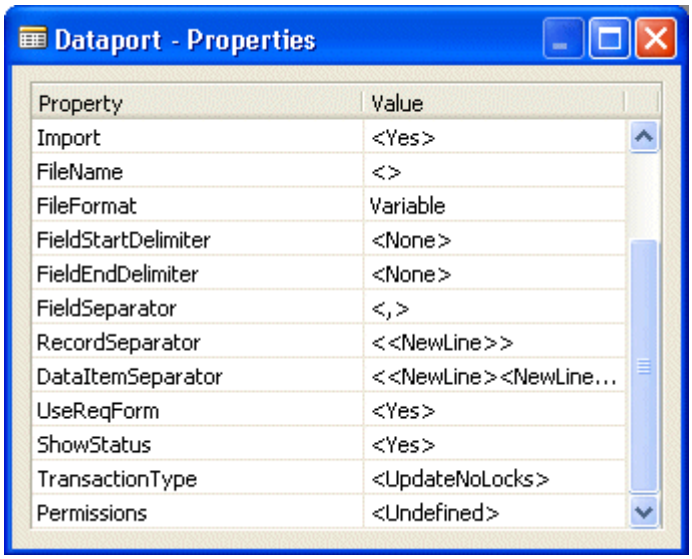
1. Установить в свойстве *FileFormat* значение Variable.

2 Установить в свойстве *FieldSeparator* значение «,» (запятая).

3. В свойствах *FieldStartDelimiter* и *FieldEndDelimiter* установить значения <None>.

После этих изменений свойства датапорта должны выглядеть так:

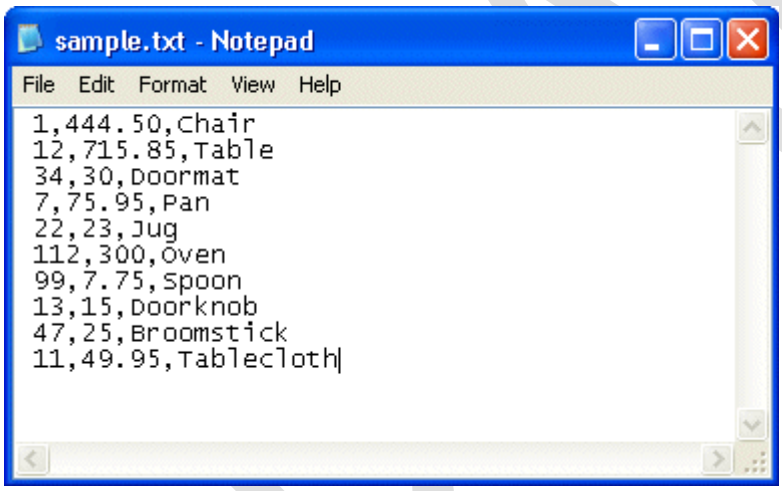
DRAFT



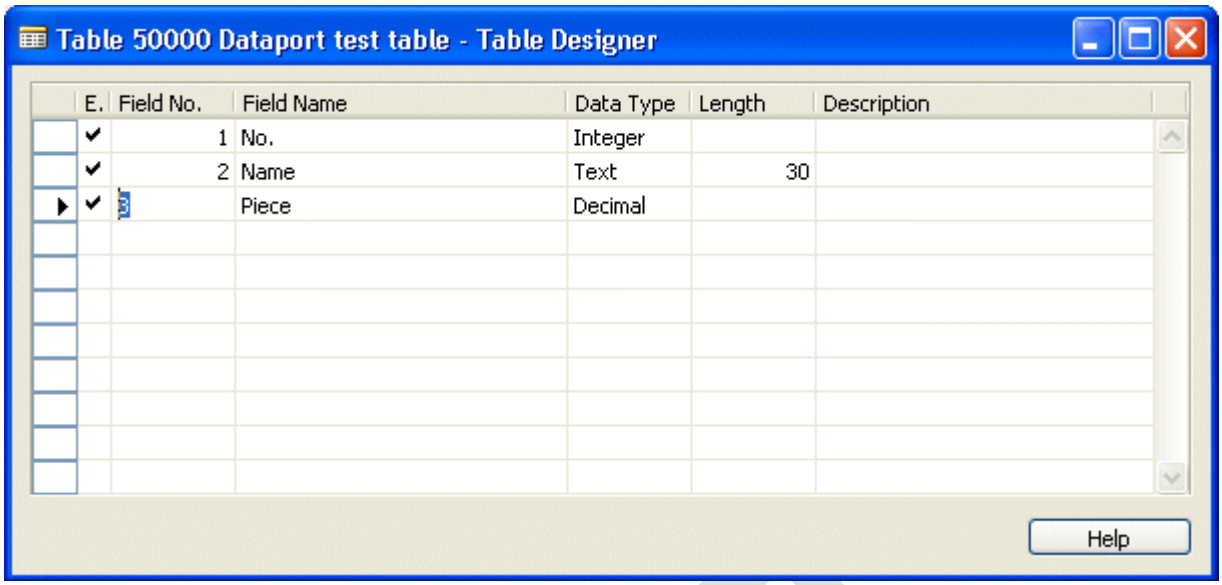
Это единственные отличия между датапортом для фиксированного формата и датапортом для переменного формата. Сохраните и откомпилируйте датапорт. Результаты запуска данного датапорта будут такими же, как и для датапорта для фиксированного формата.

> **Совет**
>
> Если порядок полей в импортируемом файле отличается от порядка в таблице, то нужно изменить порядок полей в Field Designer согласно порядку полей в файле.

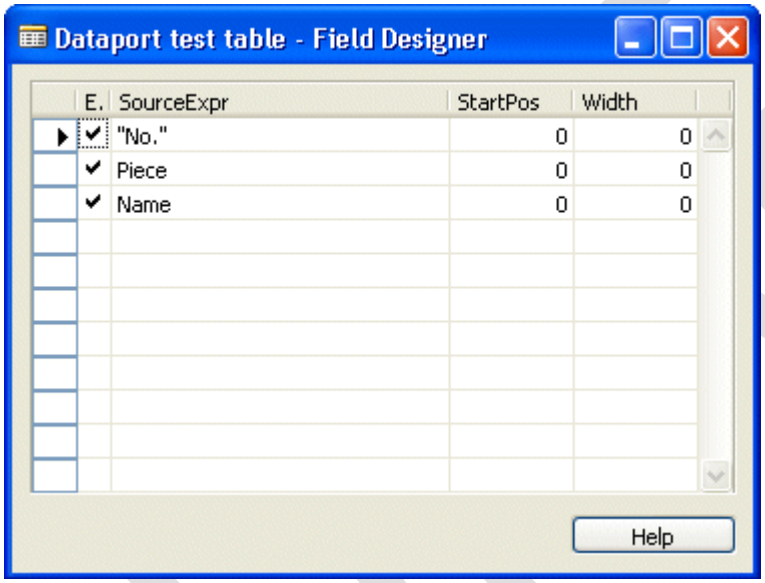
Если формат импортируемого файла такй:



А порядок полей в таблице такой (такой же, как и раньше):



В этом случае нужно в Field Designer установить такой же порядок полей, как в импортируемом файле:



Так как поля из **Field Menu** добавляются по одному (а не все сразу), значение полей **StartPos** и **Width** не рассчитываются. На этом можно не обращать внимания, так как они не используются в датапортах для переменных форматов.

Импорт или Экспорт: Динамический Датапорт

Последний пример будет чуть сложнее предыдущих. В нем будет создан датапорт, который будет использоваться для обновления цен в таблице ***Item*** с помощью внешнего приложения Microsoft Excel.

Датапорт будет работать следующим образом:

1. В форме запроса пользователь должен будет указать, хочет он выполнить импорт или экспорт, а также указать имя файла.

2. При экспорте, обрабатываются записи, у которых в поле **Gen. Prod. Posting Group** установлено значение РОЗНИЦА.

3. Кроме того, экспортируются значения, у которых значение поле **No.** принадлежит диапазону от 1000 до 2000.

4. Экспортируются только три поля: **No.**, **Description** и **Unit Price**.

5. Дополнительно экспортируется поле **Description** (text) из таблицы ***Tariff Number,*** связанной с записью через поле **Tariff No**. Таким образом, текст из поля **Description** будет четвертым полем.

6 Пользователь изменяет данные поля **Unit Price** во внешнем приложении (по крайней мере, у него будет такая возможность). Это означает, что информация в данном поле может разливаться в базе данных и внешнем файле. Поэтому значение поля **Unit Price** из внешнего файла должно быть снова загружено в базу данных. Т.е. записи базы данных должны обновляться при импорте файла.

> **Примечание**
>
> Одной из причин фильтрации по полю **No.** является исключение записей с запятыми в поле **Description**. Это позволит упростить работу с данным примером в Excel.

Создание Части, Выполняющей Экспорт

Начнем с создания части датапорта, выполняющей экспорт. Первой задачей является определение формата внешнего файла. Так как файл будет редактироваться в Excel, формат должен поддерживаться. Текстовый файл с разделителями (точка с запятой) будет лучшим выбором.

Создадим датапорт:

1. Открыть **Object Designer** и щелкнуть кнопку **Dataport**, затем кнопку **New**, чтобы создать новый датапорт.

2. Открыть окно ***Properties*** (SHIFT+F4) для датапорта и установить следующие свойства:

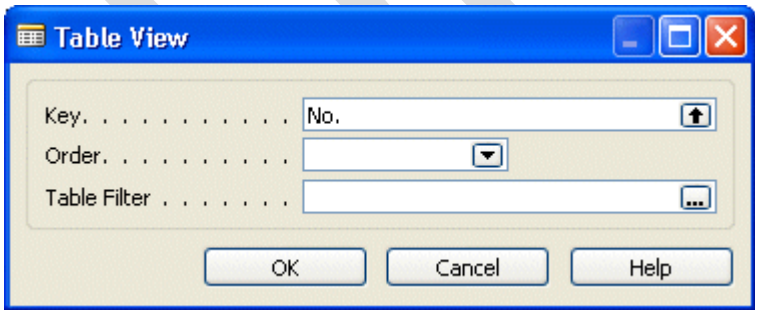
Property	Value
FileFormat	Variable – значение по умолчанию
FieldStartDelimiter	" (двойные кавычки) – значение по умолчанию
FieldEndDelimiter	" (двойные кавычки) – значение по умолчанию
FieldSeparator	; (точка с запятой)

Для остальных свойств оставьте значения по умолчанию.

3. Создайте элемент данных, на основании таблицы ***Item***.

4. Откройте окно ***Properties*** для элемента данных Item.

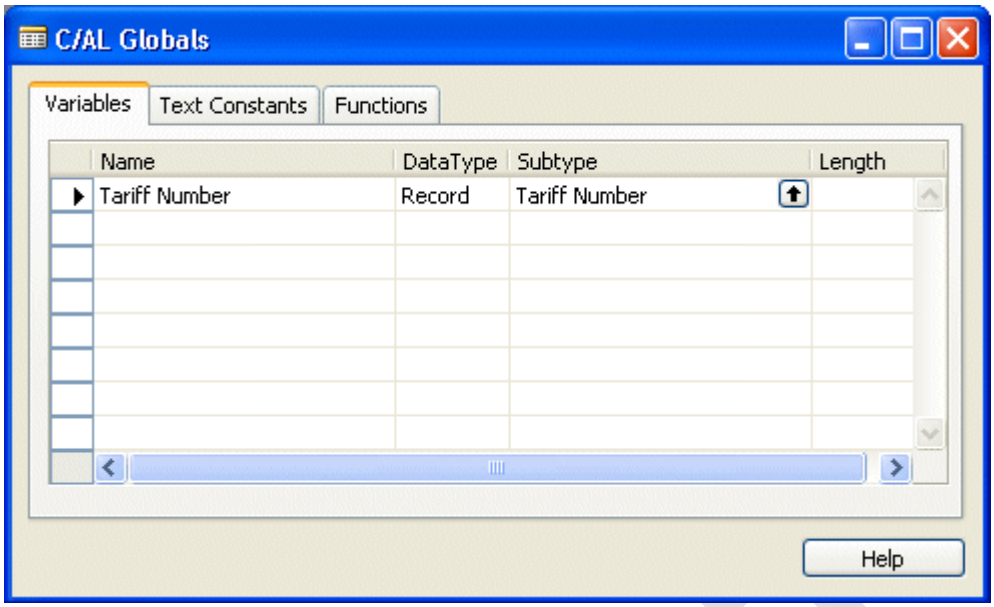
5. Щелкните кнопку AssistButton в поле *DataItemTableView*, чтобы открыть окно ***Table View***.



6. Установите в поле **Key** значение **No.** (можно выбрать в окне ***Key List***).

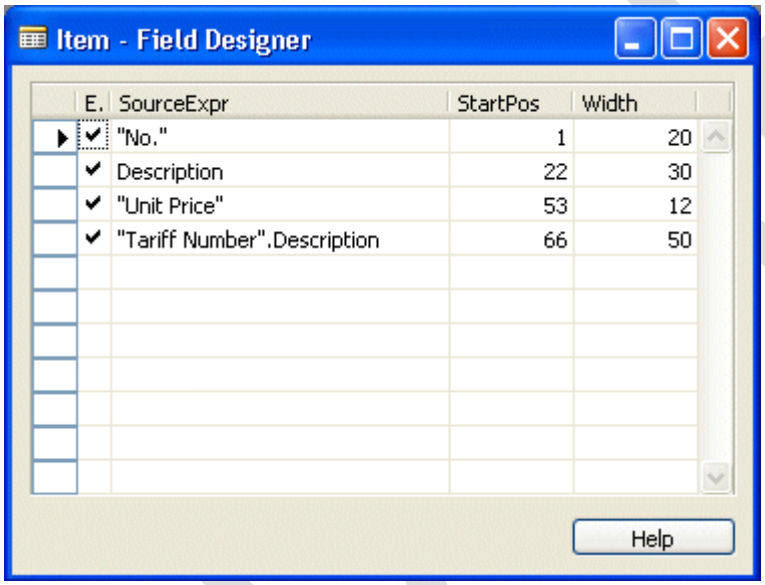
Не используйте поле **Table Filter** для установки фильтров. Причины этого станут вам понятны позднее.

7. Щелкните Вид, C/AL Globals, чтобы открыть окно ***C/AL Globals*** и создайте глобальную переменную Tariff Number с типом данных Record, и ***Tariff Number*** в качестве подтипа:



8. Выберите элемент данных Item, затем щелкните Вид, Dataport Fields, чтобы открыть окно ***Field Designer***.

9. Настройте поля датапорта как показано ниже:



Используйте кнопку AssistButton в поле **SourceExpr,** чтобы выбрать поля из таблицы ***Item***.

Однако, вам придется ввести название поля из таблицы ***Tariff Number*** вручную. Используйте такой же формат как на рисунке выше.

10. Чтобы установить фильтр, на основании которого будут отбираться записи со значением поля No. от 1000 до 2000 и значением поля **Gen. Prod. Posting Group** = РОЗНИЦА, откройте C/AL editor и введите следующий код в триггер *OnPreDataItem* элемента данных Item:

```
IF NOT CurrDataport.IMPORT THEN BEGIN
  Item.SETRANGE("Gen. Prod. Posting Group",'RETAIL');
  Item.SETRANGE("No.",'1000','2000');
END;
```

О фильтрах

Фильтры устанавливаются только в том случае, если датапорт используется для экспорта данных (помним, что пользователь решает, будет выполняться импорт или экспорт динамически). Причина для использования такой конструкции вместо настройки фильтров таблицы в свойстве *DataItemTableView* заключается в следующем. Если используется TableFilter, то фильтр устанавливается всегда, даже в том случае, когда датапорт импортирует данные. Однако, так как поле **Gen. Prod. Posting Group** не присутствует в импортируемом файле, то ни одна запись не будет загружена.

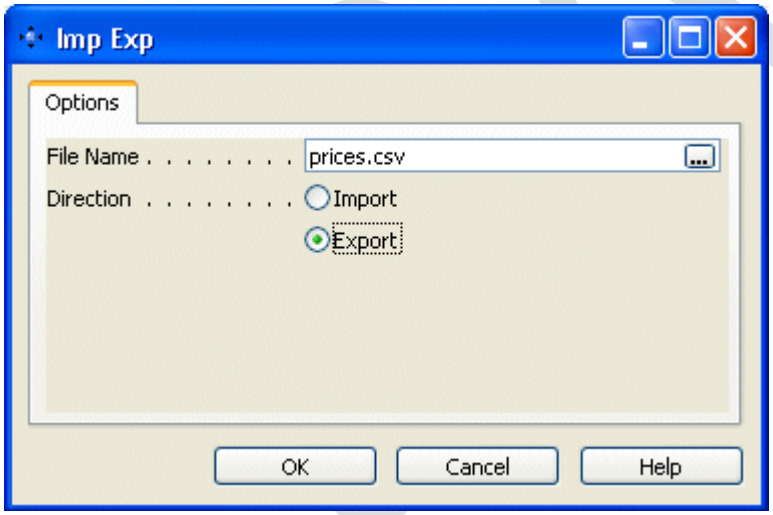
11. Чтобы получить текст и таблицы ***Tariff Number*** и экспортировать его, введите следующий код в триггер OnBeforeExportRecord элемента данных:

```
IF "Tariff No." <> '' THEN
BEGIN
  "Tariff Number"."No." := "Tariff No.";
  "Tariff Number".FIND;
END
ELSE
  "Tariff Number".Description := 'NO TARIFF NUMBER';
```

Этого достаточно, чтобы создать часть датапорта, выполняющую экспорт записей.

12. Сохраните и откомпилируйте датапорт.

Когда вы запустите датапорт, появится форма запроса, которую следует заполнить следующим образом:



Когда вы щелкните кнопку OK, датапорт запустится и все записи, соответствующие указанным критериям, будут записаны в файл prices.csv. После открытия файла в Excel он должен выглядеть следующим образом:

prices.csv

	A	B	C	D	E	F
1	1000	Bicycle	4,100	NO TARIFF NUMBER		
2	1001	Touring Bicycle	4,200	NO TARIFF NUMBER		
3	1100	Front Wheel	1,001	NO TARIFF NUMBER		
4	1110	Rim	50	NO TARIFF NUMBER		
5	1150	Front Hub	500	NO TARIFF NUMBER		
6	1200	Back Wheel	1,200	NO TARIFF NUMBER		
7	1250	Back Hub	1,100	NO TARIFF NUMBER		
8	1300	Chain Assy	800	NO TARIFF NUMBER		
9	1310	Chain	0	NO TARIFF NUMBER		
10	1700	Brake	600	NO TARIFF NUMBER		
11	1710	Hand rear wheel Brake	0	NO TARIFF NUMBER		
12	1896-S	ATHENS Desk	649.4	Desks		
13	1900-S	PARIS Guest Chair, bla	125.1	Other chairs, upholstered		
14	1906-S	ATHENS Mobile Pedest	281.4	Closets with door/drawers		
15	1908-S	LONDON Swivel Chair, b	123.3	Swivel chairs, upholstered		
16	1920-S	ANTWERP Conference T	420.4	Other office furniture		
17	1924-W	CHAMONIX Base Storag	136.4	Other office furniture		
18	1928-S	AMSTERDAM Lamp	35.6	Desk lamps		
19	1928-W	ST.MORITZ Storage Uni	342.1	Other office furniture		
20	1936-S	BERLIN Guest Chair, ye	125.1	Other chairs, upholstered		
21	1952-W	OSLO Storage Unit/She	158.5	Other office furniture		
22	1960-S	ROME Guest Chair, gre	125.1	Other chairs, upholstered		
23	1964-S	TOKYO Guest Chair, blu	125.1	Other chairs, upholstered		
24	1964-W	INNSBRUCK Storage Ur	292	Closets with door/drawers		
25	1968-S	MEXICO Swivel Chair, b	123.3	Swivel chairs, upholstered		
26	1968-W	GRENOBLE Whiteboard	974.8	Other office furniture		
27	1972-S	MUNICH Swivel Chair, y	123.3	Swivel chairs, upholstered		
28	1972-W	SAPPORO Whiteboard,	974.8	Other office furniture		
29	1976-W	INNSBRUCK Storage Ur	256.1	Closets with door/drawers		

prices

Теперь вы должны изменить цены некоторых товаров и сохранить лист как newprices.csv. Этот файл будет загружен в следующем разделе.

Создание Части, Выполняющей Импорт

Теперь создадим часть датапорта, которая может импортировать новые цены. Эту часть очень легко создать. При запуске датапорта пользователь указывает, должны ли записи импортироваться или экспортироваться, также он указывает файл, в который будет производиться запись (чтение).

Чтобы указать, что импортируемые записи должны обновлять существующие записи новыми ценами:

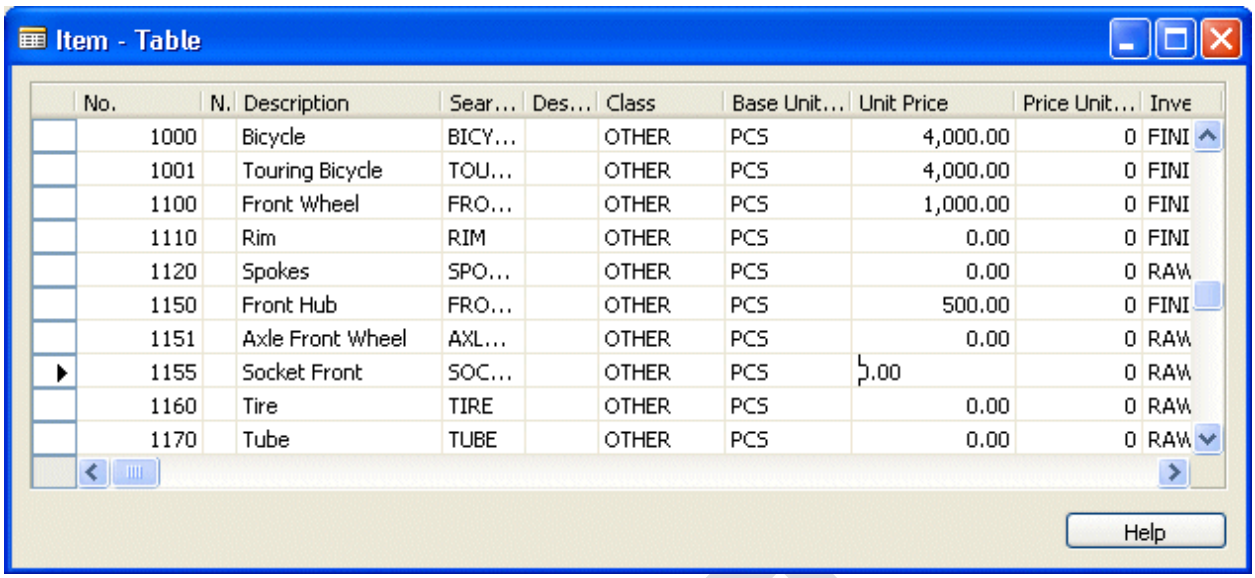
1. Откройте датапорт в Dataport Designer и откройте окно ***Properties*** для элемента данных.

2. Установите в свойстве *AutoSave* значение Да.

3. Установите в свойстве *AutoUpdate* значение Да.

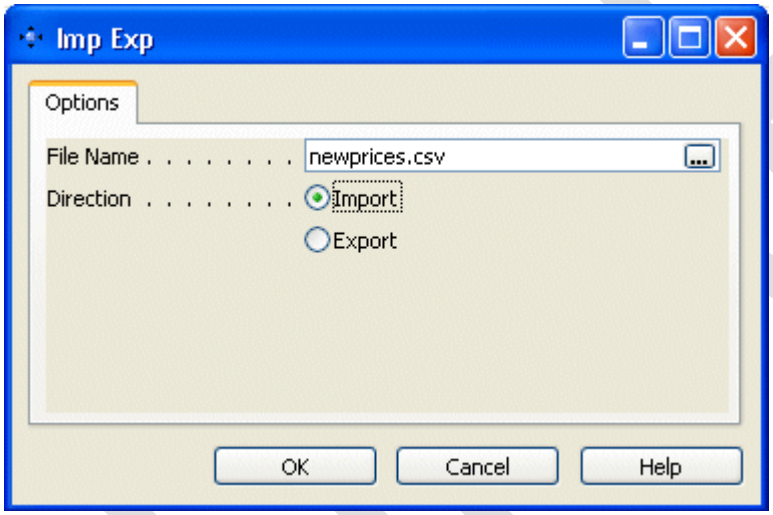
Эффект от этих установок проявляется в том, что существующие записи обновляются на основании импортируемых данных, в данном случае **Unit Price**.

Перед тем как датапорт будет использован для импорта обновленных данных, записи выглядят так:



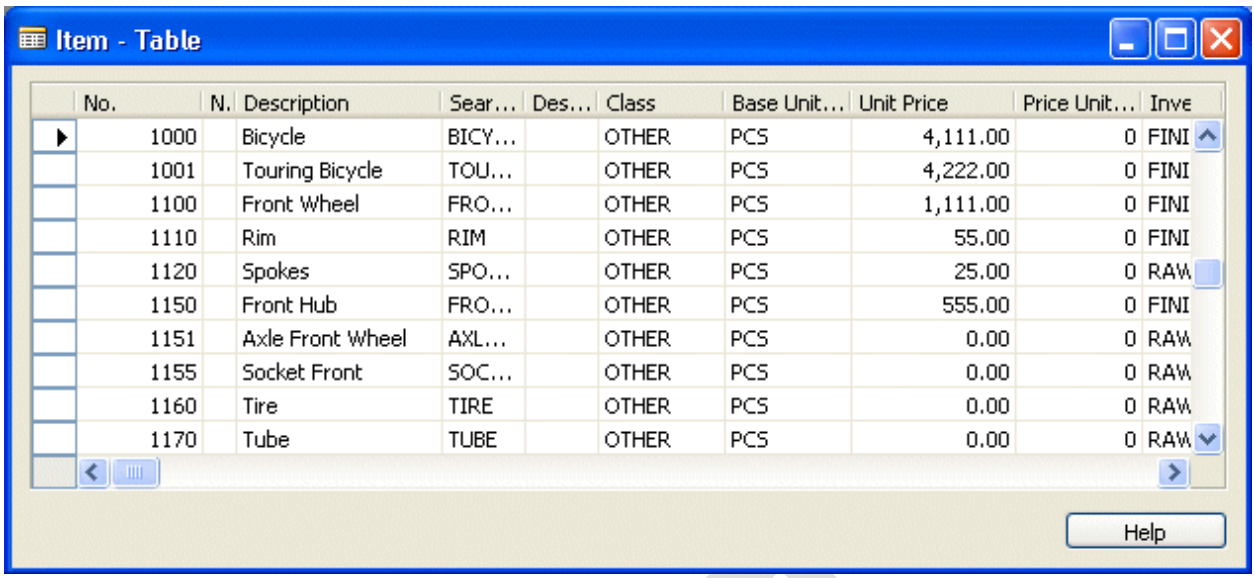
Поле **Unit Price** было перемещено так, чтобы помещаться на картинке.

Чтобы выполнить импорт данных, запустите датапорт и заполните форму запроса следующим образом:



Чтобы найти файл newprices.csv используйте кнопку AssistButton. Помните, что записи не будут сохранены в базе данных, если запустить датапорт из **Dataport Designer**. Его нужно запустить из **Object Designer**.

После того, как записи были импортированы, записи в таблице ***Item*** будут выглядеть так:



Item - Table

No.	N.	Description	Sear...	Des...	Class	Base Unit...	Unit Price	Price Unit...	Inve
1000		Bicycle	BICY...		OTHER	PCS	4,111.00	0	FINI
1001		Touring Bicycle	TOU...		OTHER	PCS	4,222.00	0	FINI
1100		Front Wheel	FRO...		OTHER	PCS	1,111.00	0	FINI
1110		Rim	RIM		OTHER	PCS	55.00	0	FINI
1120		Spokes	SPO...		OTHER	PCS	25.00	0	RAW
1150		Front Hub	FRO...		OTHER	PCS	555.00	0	FINI
1151		Axle Front Wheel	AXL...		OTHER	PCS	0.00	0	RAW
1155		Socket Front	SOC...		OTHER	PCS	0.00	0	RAW
1160		Tire	TIRE		OTHER	PCS	0.00	0	RAW
1170		Tube	TUBE		OTHER	PCS	0.00	0	RAW

Help

Обратите внимание, что первые шесть товаров были изменены.

DRAFT

ЧАСТЬ 8 XMLPORT

ГЛАВА 21 XMLPORT

Концепция XMLport близка концепции датапортов, так как объекты данного типа также используются для экспорта и импорта данных. Различие заключается в использовании формата XML. Это позволяет упорядочить обмен информацией между различными учетными системами.

- XMLport – базовые понятия
- Создание XMLport
- Примеры XMLport
- Проверка данных
- XMLport и Business Notifications

21.1 XMLport – Базовые Понятия

XMLport позволяет импортировать данные, полученные в формате XML, в базу данных Dynamics NAV и экспортировать данные в формате XML из базы данных Dynamics NAV. Чтобы создавать XMLport от вас потребуются базовые знания XML.

Создание XMLport осуществляется в XMLport Designer, который открывается из Object Designer.

Сохранение, Компиляция и Запуск XMLport

После завершения работы над XMLport его нужно сохранить и откомпилировать, прежде чем его можно будет запускать. Обычно это делается после окончания работ над XMLport. Однако XMLport можно сохранить, даже если работы над ним еще ведутся (из-за чего его нельзя откомпилировать). Кроме того, можно выполнить тестовую компиляцию не сохраняя и не закрывая XMLport.

Сохранение и Закрытие XMLport

XMLport закрывается при закрытии XMLport Designer.

Чтобы сохранить XMLport:

1. При закрытии XMLport, среда C/SIDE спросит, требуется ли сохранить изменения. Если это новый XMLport (XMLport, который еще ни разу не сохранялся), то нужно будет указать имя и идентификатор (ID). Идентификатор должен быть уникальным и соответствовать правилам присваивания номеров объектам. Для дополнительной информации об идентификаторах объектов обратитесь к сертифицированному партнеру Microsoft Business Solutions.

Примечание

DRAFT

Если в свойствах XMLport заполнить свойства ID и Name, то будут использоваться эти значения и система не предложит ввести ID и Name при закрытии XMLport.

2 Значение поля опции **Compiled** по умолчанию - TRUE (отображается как флажок). Если ваш XMLport еще не готов к компиляции, снимите флажок, щелкнув в поле.

3. Щелкните OK, чтобы сохранить XMLport.

Вы можете сохранить XMLport, не закрывая его, выбрать пункт Save или Save As в меню Файл. Используя Save As, вы можете переименовать существующий XMLport (помните, что вы можете скопировать XMLport, открыв и сохранив его под новым именем).

Компиляция XMLport

XMLport, подобно другим объектам в среде C/SIDE, должны компилироваться до того, как их можно будет запускать. Как описано выше, вы можете выбрать возможность компиляции XMLports тогда, когда сохраняете его.

Во время создания XMLport, возможно, потребуется произвести его тестовую компиляцию, чтобы найти возможные ошибки. Для этого нужно щелкнуть Сервис, Компилировать (F11).

Запуск XMLport

XMLport запускаются из программных модулей. При создании XMLport часто его требуется запускать до того как он будет интегрирован в приложение, для этого нужно запускать тестовый программный модуль, который будет вызывать XMLport и создавать потоки для экспорта или импорта данных из файла. Ниже приведен пример тестового программного модуля, который запускает XMLport для импорта данных их XML файла:

```
Codeunit 74300 - Import - C/AL Editor
Documentation()

OnRun()
TestFile.OPEN('C:\XML-Test\XML Import\Import.xml');
TestFile.CREATEINSTREAM(TestStream);
XMLPORT.IMPORT(74300,TestStream);
TestFile.CLOSE;
```

Четыре функции в триггере OnRun выполняют следующие действия:

- Открывают файл Import.xml.
- Создают объект InStream, для получения данных в формате XML из файла.
- Загружают определенный XMLport и указывает ему источник, из которого он должен читать и разбирать входящие данные в формате XML.
- Закрывают файл Import.xml.

Информацию о потоке данных из/в файлов и из/в компонентов automation, можно получить в онлайновых справках *C/SIDE Reference Guide* и *Development Guide for Communication Components*. Последняя доступна на DVD-диске и устанавливается вместе с NAV Application Server.

21.2 Создание XMLport

Документ XML содержит теги XML, которые идентифицируют природу содержащихся в нем данных. Чтобы создать XMLport, импортирующий данные из документа XML, нужно определить все теги XML и указать их тип, также нужно определить является ли тег элементом или атрибутом. После это для тегов устанавливается соответствие со структурами данных (таблицы, записи, поля) в базе данных Dynamics NAV. При запуске объекта XMLport он обработает входящий документ XML: прочитает входящий поток данных и выполнит операции в базе данных.

Если требуется создать XMLport, которые экспортирует данные из базы данных Dynamics NAV в формате XML, то XMLport Designer позволяет задать структуру тегов документа XML и установить соответствие с данными в базе данных. При запуске объекта XMLport он прочитает требуемые данные из базы данных, добавит необходимые теги XML в документ XML и запишет документ в исходящий поток данных.

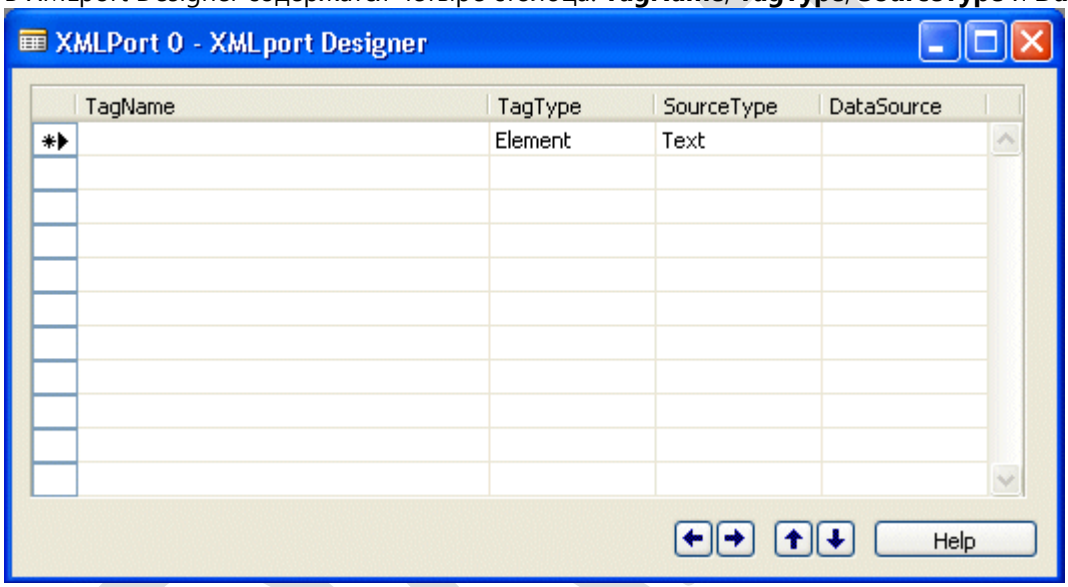
XMLport не обрабатывают документы XML которые:

- Изменяют существующие данные в базе данных.
- Ищут и удаляют записи в базе данных.
- Делают запросы на получение данных (например, каталог товаров).

Если вам нужно работать с входящими документами указанных типов для операций с базой данных и достижения требуемого результата следует использовать C/AL код.

XMLport Designer

В XMLport Designer содержатся четыре столбца: **TagName**, **TagType**, **SourceType** и **DataSource**.



В данные поля следует вводить информацию следующих типов:

TagName

В поле **TagName** вводится название тега XML, независимо того является ли он элементом или атрибутом. Нужно вводить теги в той последовательности, как они появляются в документе XML. Родительский элемент должен предшествовать дочерним элементам. Для дочерних элементов устанавливается отступ на один уровень, относительно их родительских элементов. Атрибуты перечисляются под элементами, для которых они определены, для них также устанавливается отступ.

TagType

Чтобы указать, является ли тэг, определенный в поле **TagName,** элементом или атрибутом используется поле **TagType.** В выпадающем списке для поля **TagType** содержится два варианта: Element и Attribute. Element является значением по умолчанию.

SourceType

В данном поле указывается соответствующая структура в базе данных Dynamics NAV. В выпадающем списке для поля **SourceType** содержится три варианта: Text, Table и Field. Text является значением по умолчанию.

Text: Данная опция выбирается, когда данные XML невозможно напрямую связать с базой данных, либо же когда эти данные в базе данных не нужны. Значение поля **Text** будет помещено в переменную типа text, имя, которой указано в свойстве VariableName (по умолчанию туда записывается название тега). С данной переменной можно работать как с обычной переменной определенной в окне C/AL. Также можно использовать переменную типа bigtext, для этого в свойстве TextType нужно установить значение BigText.

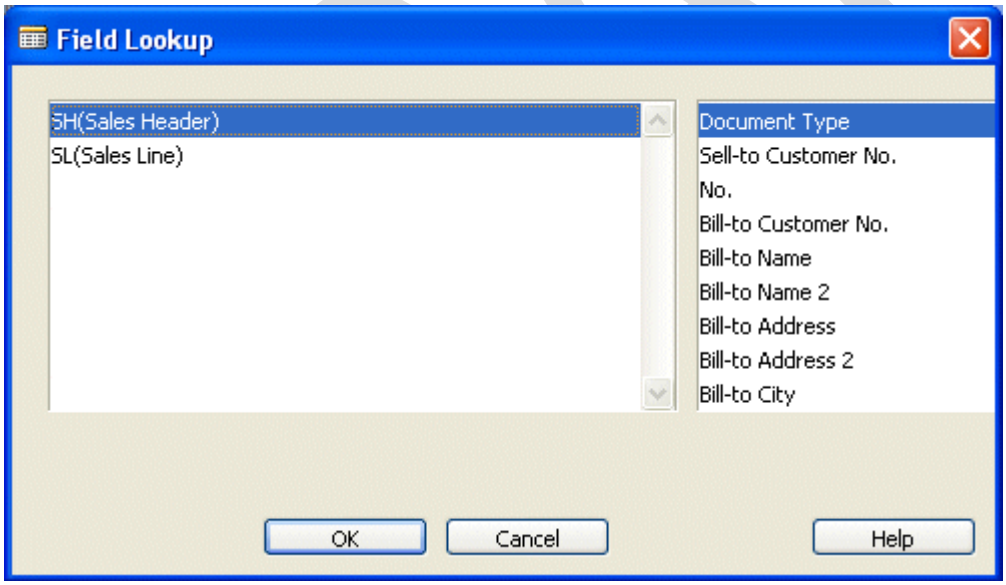
Table: Данная опция выбирается, чтобы указать связь тэга с таблицей. Как и в случае с вариантом Text можно указать имя переменной для таблицы, данная переменная будет доступна как обычная глобальная переменная типа record. По умолчанию в качестве имени переменной используется наименование имени таблицы.

Field: Данная опция выбирается, чтобы указать связь тега с полем. Чтобы использовать данную опцию, вначале нужно объявить в качестве родительского элемента таблицу. Иначе при попытке компиляции объекта XMLport возникнет ошибка.

Поле **DataSource** связано с полем **SourceType** следующим образом:

- Если SourceType = *Table*, то при щелчке по кнопке AssistButton в поле **DataSource** откроется окно ***Table List***. Также можно выбрать таблицу с помощью свойства SourceTable. Если для таблицы определено название переменной, то в поле **DataSource** будет использоваться следующий формат tablevariablename(tablename). Если переменная для таблицы не задана, то будет использоваться формат <tablename>(tablename).
- Если SourceType = *Field*, то при щелчке по кнопке AssistButton в поле **DataSource** откроется окно ***Field Lookup***. В данном окне можно выбрать поле одной из таблиц, определенных в XMLport. Формат отображаемого значения будет tablevariablename::fieldname.

Ниже приведен пример окна ***Field Lookup***, в котором можно выбрать поля из таблиц ***Sales Header*** и ***Sales Line***:



- Если SourceType = *Text*, то в поле **DataSource** будет отображаться наименование текстовой переменной (или название тега, если имя переменной не задано).

Свойства, Функции и Триггеры XMLport

Для работы с XMLport существует ряд свойств, функций и триггеров. Свойства, функции и триггеры есть как на уровне объекта в целом, так и на уровне элементов (Field, Table и Text). Доступные свойства зависят от выбора, сделанного в полях **TagType** и **SourceType** в XMLport Designer. Описания всех свойств, функций и триггеров XMLport в онлайновой справке *C/SIDE Reference Guide*.

21.3 Примеры использования XMLport

В данном разделе будет приведено несколько примеров использования XMLport для импорта данных из документов XML в базу данных Dynamics NAV.

Пример 1

В данном примере будет создан XMLport для импорта данных о заказах продажи из документа XML - XML Sales Order.xml, в базу данных Dynamics NAV. Заказ продажи показан ниже:


```
  <?xml version="1.0" ?>
- <HeaderLines>
  - <SalesOrder>
    - <Header Date="17-02-04" Type="Quote">
      - <SellTo Country="GB">
          <Name>The Canon Group PLC</Name>
          <Address>192 Market Square</Address>
          <City>Birmingham</City>
          <Zip>B27 4KT</Zip>
        </SellTo>
      - <BillTo Country="DK">
          <Name>The Cannon Group PLC</Name>
          <Address>Frydenlunds Alle</Address>
          <City>Vedbaek</City>
          <Zip>2950</Zip>
        </BillTo>
      - <Lines>
        - <Item PartNum="LS-150">
            <ProductName>Loudspeaker, Cherry, 150W</ProductNa
            <Quantity>8</Quantity>
            <UnitPrice>129,00</UnitPrice>
            <ShipmentDate />
            <Comment>Confirm the voltage is 75W</Comment>
          </Item>
        - <Item PartNum="LS-MAN-10">
            <ProductName>Manual for Loudspeakers</ProductName:
            <Quantity>20</Quantity>
            <UnitPrice />
            <ShipmentDate />
            <Comment />
          </Item>
        - <Item PartNum="LS-2">
            <ProductName>Cables for Loudspeakers</ProductName:
            <Quantity>10</Quantity>
            <UnitPrice>21,00</UnitPrice>
            <ShipmentDate />
            <Comment />
          </Item>
        </Lines>
        <Contact>Mr. Andy Teal</Contact>
        <Terms>14 days</Terms>
      </Header>
    </SalesOrder>
```


После анализа документа XML, было определено, что данные принадлежат двум таблицам: ***XML Header - Import*** и ***XML Items - Import***. Далее будет создан XMLport, который вставляет данные в эти таблицы.

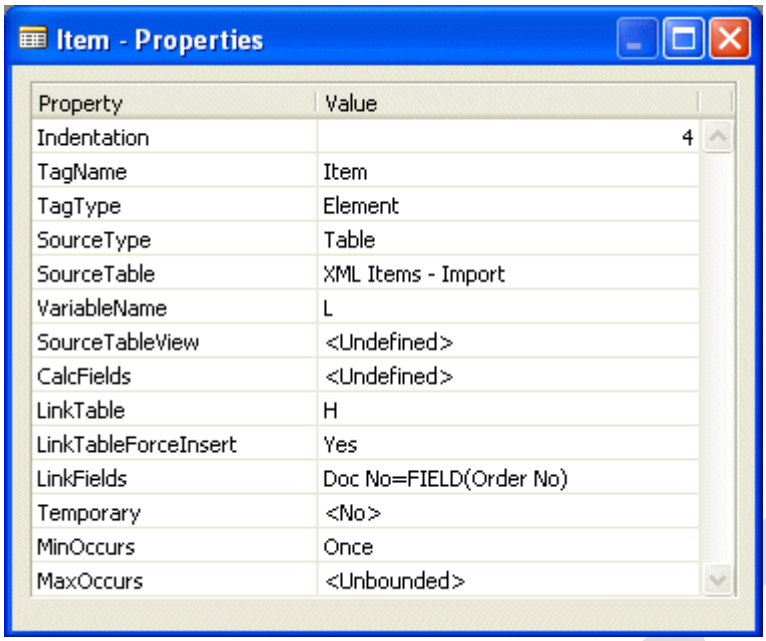
Структура XMLport следующая:

XMLport 74300 Import XML Sales Order - XMLport Designer

TagName	TagType	SourceType	DataSource
HeaderLines	Element	Text	<HeaderLines>
SalesOrder	Element	Text	<SalesOrder>
Header	Element	Table	H(XML Header - Import)
Date	Attribute	Field	H::Date
Type	Attribute	Field	H::Order Type
SellTo	Element	Text	<SellTo>
Country	Attribute	Text	SellTo Country
Name	Element	Field	H::SellTo Name
Address	Element	Field	H::SellTo Address
City	Element	Field	H::SellTo City
Zip	Element	Field	H::SellTo ZIP
BillTo	Element	Text	<BillTo>
Country	Attribute	Text	BillTo Country
Name	Element	Field	H::BillTo Name
Address	Element	Field	H::BillTo Address
City	Element	Field	H::BillTo City
Zip	Element	Field	H::BillTo ZIP
Lines	Element	Text	<Lines>
Item	Element	Table	L(XML Items - Import)
PartNum	Attribute	Field	L::Part Number
ProductName	Element	Field	L::Product Name
Quantity	Element	Field	L::Quantity
UnitPrice	Element	Field	L::Unit Price
ShipmentDate	Element	Field	L::ShipDate
Comment	Element	Field	L::Comment
Contact	Element	Field	H::Contact Person
Terms	Element	Field	H::Payment Terms

Help

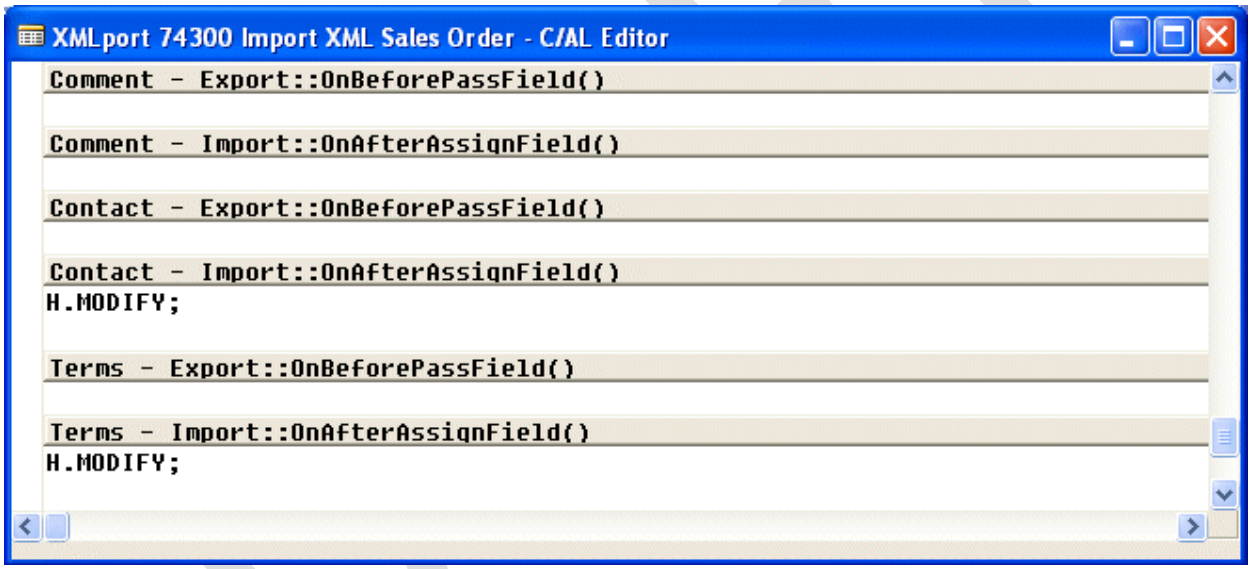
Структура XMLport показывает, как для данных документа XML устанавливается соответствие с таблицами базы данных, между которыми есть связь «заголовок-строки». XMLport должен вставить информацию в заголовок, а затем перейти к вставке информации из строк. Чтобы гарантировать такую последовательность, для тега <Item> было установлено в свойстве LinkedTableForcedInsert значение Да. Чтобы указать связь между таблицами ***XML Header - Import*** и ***XML Items - Import*** также использовались свойства данного тега: LinkTable и LinkFields. Очень важно указать данную связь, чтобы обеспечить вставку данных в правильные таблицы. Обратите внимание, для таблицы ***XML Items - Import*** используется переменная "L", а для таблицы ***XML Header - Import*** переменная "H". Окно ***Property*** для тега <Item> выглядит следующим образом:



Item - Properties

Property	Value
Indentation	4
TagName	Item
TagType	Element
SourceType	Table
SourceTable	XML Items - Import
VariableName	L
SourceTableView	<Undefined>
CalcFields	<Undefined>
LinkTable	H
LinkTableForceInsert	Yes
LinkFields	Doc No=FIELD(Order No)
Temporary	<No>
MinOccurs	Once
MaxOccurs	<Unbounded>

После того как все записи будут вставлены в таблицу ***XML Items - Import*** нужно будет обновить поля **Contact Person** и **Payment *Terms*** таблицы ***XML Header - Import***. Это будет выполнено с помощью следующего кода на языке C/AL:



XMLport 74300 Import XML Sales Order - C/AL Editor

```
Comment - Export::OnBeforePassField()

Comment - Import::OnAfterAssignField()

Contact - Export::OnBeforePassField()

Contact - Import::OnAfterAssignField()
H.MODIFY;

Terms - Export::OnBeforePassField()

Terms - Import::OnAfterAssignField()
H.MODIFY;
```

Example 2

В данном примере показано, как обрабатывать ситуацию, когда документ XML содержит несколько значений, которые в базе данных Dynamics NAV представлены одной структурой.

В базе данных есть таблица ***XML Sales Lines,*** которая содержит одно поле **Shipment** с типом данных DateTime. Однако в документе XML, содержащем информацию о строках заказа продажи данное значение представлено двумя тегами: <ShipDate> и <ShipTime>:

```
  <?xml version="1.0" encoding="utf-8" ?>
- <SalesOrder>
  - <OrderHeader OrderDate="17-09-04">
    - <SellTo Country="GB">
        <Name>The Cannon Group PLC</Name>
        <Address>192 Market Square</Address>
        <City>Birmingham</City>
        <Zip>GB-CV6 1GY</Zip>
      </SellTo>
    - <BillTo Country="DK">
        <Name>Jens Ole</Name>
        <Address>Frydenlunds Alle 6</Address>
        <City>Vedbaek</City>
        <Zip>DK-2950</Zip>
        <PaymentTerms>14days</PaymentTerms>
      </BillTo>
    </OrderHeader>
  - <Items>
    - <Item PartNum="LS-75">
        <ProductName>Loudspeaker, Cherry, 75W</ProductName>
        <Quantity>10</Quantity>
        <UnitPrice>79</UnitPrice>
        <ShipDate>12-10-04</ShipDate>
        <ShipTime>09:50</ShipTime>
        <Comment>Confirm the voltage is 75w</Comment>
      </Item>
    - <Item PartNum="1908-S">
        <ProductName>LONDON Swivel Chair, blue</ProductName>
        <Quantity>12</Quantity>
        <UnitPrice>190,926</UnitPrice>
        <ShipDate>12-10-04</ShipDate>
        <ShipTime>09:50</ShipTime>
      </Item>
    </Items>
  </SalesOrder>
```


Одним из вариантов решения данной проблемы является кэширование значений из тегов <ShipDate> и <ShipTime>, затем преобразование этих значений в значение типа DateTime, далее нужно выполнить присваивание преобразованного значения полю **Shipment** таблицы ***XML Sales Lines***.

В XMLport Designer нужно определить два тега <ShipDate> и <ShipTime>, для обоих установить в поле SourceType значение Text:

Tag Name	Tag Type	Source Type	Data Source
ShipDate	Element	Text	<ShipDate>
ShipTime	Element	Text	<ShipTime>

В триггере OnAfterAssignVariable тега <ShipTime> нужно написать код, который преобразует две текстовые переменные в одну переменную DateTime и присвоит полученное значение полю **Shipment** таблицы ***XML Sales Lines***.

Пример 3

В деловых документах обычно не требуется указывать одну информацию несколько раз. Однако в реляционных базах данных, одна информация повторяется в различных таблицах и используется для создания связей. В следующем документе XML, содержащем заказ продажи, у тега <Items> есть атрибут Type="Order":

```
  <?xml version="1.0" ?>
- <SalesOrder>
  - <OrderHeader OrderDate="17-01-04">
    - <SellTo Country="GB">
        <Name>The Cannon Group PLC</Name>
        <Address>192 Market Square</Address>
        <City>Birmingham</City>
        <Zip>GB-CV6 1GY</Zip>
      </SellTo>
    - <BillTo Country="DK">
        <Name>Jens Ole</Name>
        <Address>Frydenlunds Alle 6</Address>
        <City>Vedbaek</City>
        <Zip>DK-2950</Zip>
        <PaymentTerms>14days</PaymentTerms>
      </BillTo>
    </OrderHeader>
  - <Items Type="Order">
    - <Item PartNum="LS-75">
        <ProductName>Loudspeaker, Cherry, 75W</ProductName>
        <Quantity>10</Quantity>
        <UnitPrice>79</UnitPrice>
        <Comment>Confirm the voltage is 75w</Comment>
      </Item>
    - <Item PartNum="1908-S">
        <ProductName>LONDON Swivel Chair, blue</ProductName>
        <Quantity>12</Quantity>
        <UnitPrice>190,926</UnitPrice>
        <ShipDate>12-02-04</ShipDate>
      </Item>
    </Items>
  </SalesOrder>
```

Проблема заключается в том, что информация Type="Order" должна записываться в поле **Document Type** соответствующей таблицы, в данном примере это ***XML Sales Lines***. Поэтому нужно кэшировать значение "Order".

Tag Name	Tag Type	Source Type	Data Source
Type	Attribute	Text	Type

А для выполнения присваивания значения Type каждой записи таблицы ***XML Items - Import*** нужно добавить следующий код в триггер Import::OnAfterInitRecord тега <Item>:

```
Item - Import::OnAfterInitRecord()
      L.Document Type := Type;
```

Пример 4

Иногда входящий документ XML может не содержать достаточно информации для вставки записи. Чтобы решить эту проблему нужно выполнить предварительное присваивание значений полям, для

которых не достаточно данных. Это выполняется в триггерах Import::OnAfterInitRecord и Import::OnBeforeInsertRecord.

Например, в полученном документе XML, содержащем заказ продажи, нет информации о номере строки (Line No.) поэтому нужно выполнить следующее присваивание:

```
Item - Import::OnBeforeInsertRecord()
If not SL.Line No. then
  SL.Line := 1001;
```

21.4 Проверка Данных

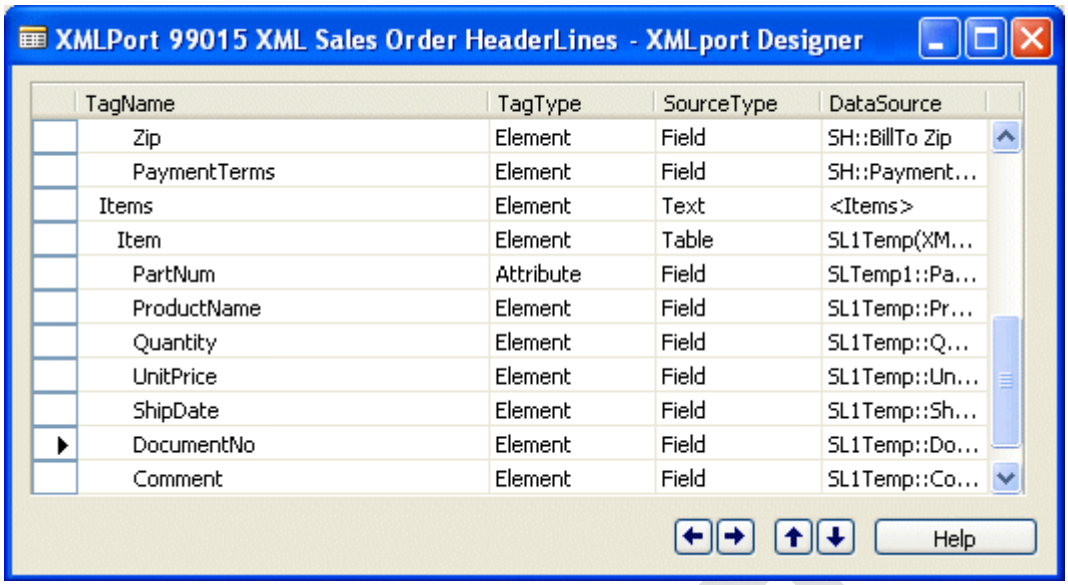
Проверка данных (validation) - основа обеспечения целостности информации, используемой приложением. Когда XMLport записывает данные в таблицу Dynamics NAV, важно избежать вставки некорректных данных. Для этого у объектов XMLport в свойстве DefaultFieldsValidation по умолчанию установлено значение Да. Однако вставка данных выполняется в той последовательности, как указаны теги XML в XMLport, а не согласно последовательности проверки в таблице. Так как последовательность вставки нарушает последовательность проверки, то ваша задача либо изменить порядок тегов XML, либо сделать так, чтобы данные не сразу вставлялись в базу данных.

Можно создать XMLport таким образом, чтобы данные записывались во временную таблицу. А затем написать код по вставке данных в соответствующую таблицу в правильном порядке. Например, документ XML, содержащий заказ продажи, может включать, так <DocumentNo> как показано ниже:

```
- <Items>
  - <Item PartNum="LS-75">
      <ProductName>Loudspeaker, Cherry, 75W</ProductName>
      <Quantity>10</Quantity>
      <UnitPrice>79</UnitPrice>
      <Comment>Confirm the voltage is 75w</Comment>
    </Item>
  - <Item PartNum="1908-S">
      <ProductName>LONDON Swivel Chair, blue</ProductName>
      <Quantity>12</Quantity>
      <UnitPrice>190,926</UnitPrice>
      <ShipDate>12-05-04</ShipDate>
      <DocumentNo>9999</DocumentNo>
    </Item>
  </Items>
```

Значение этого тега содержит номер документа, и должно быть записано в таблицу базы данных ***XML Sales Header***, предварительно значение должно быть проверено. Однако XMLport не будет работать, так как позиция тега <DocumentNo> нарушает порядок выполнения проверки в таблице ***XML Sales Lines***.

Чтобы решить проблему можно выполнить присваивание значений тегов XML, связанных с таблицей ***XML Sales Lines***, временной таблице:



TagName	TagType	SourceType	DataSource
Zip	Element	Field	SH::BillTo Zip
PaymentTerms	Element	Field	SH::Payment...
Items	Element	Text	<Items>
Item	Element	Table	SL1Temp(XM...
PartNum	Attribute	Field	SLTemp1::Pa...
ProductName	Element	Field	SL1Temp::Pr...
Quantity	Element	Field	SL1Temp::Q...
UnitPrice	Element	Field	SL1Temp::Un...
ShipDate	Element	Field	SL1Temp::Sh...
DocumentNo	Element	Field	SL1Temp::Do...
Comment	Element	Field	SL1Temp::Co...

Далее нужно написать код, который выполнит присваивание значения из тега <DocumentNo> корректной таблице. Это означает, что значение <DocumentNo> должно быть скопировано из временной таблицы в настоящую до того, как остальные поля начнут копироваться в таблицу ***XML Sales Lines***.

Использование временных таблиц позволит решить большинство проблем возникающих, когда порядок тегов в документе XML нарушает порядок выполнения проверки данных в таблице базы данных.

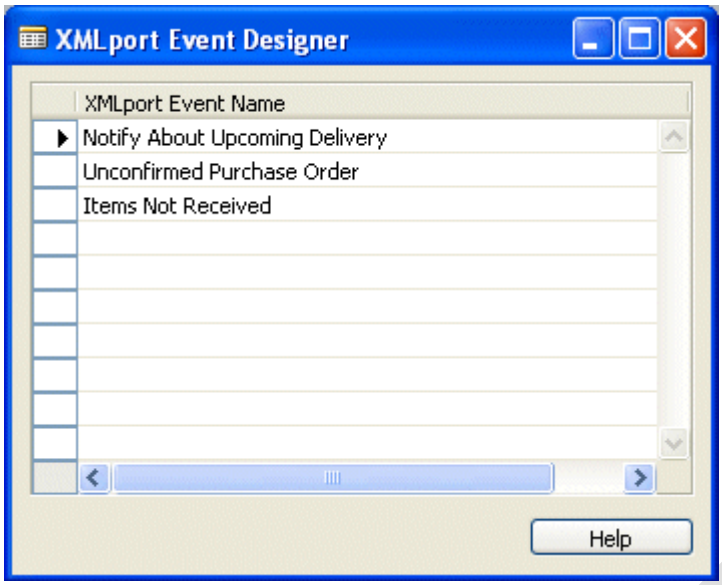
21.5 XMLport и Business Notifications

При использовании Microsoft Business Notifications для Dynamics NAV, можно использовать события XMLport. Пользователи, у которых работает Business Notifications и у которых есть подписка на эти события, получат уведомления. Например, можно создать событие "Sales Order Created" для Sales Order XMLport. События XMLport, вызванные из программного модуля, создают события в Business Notifications. Затем XMLport отправляет документ XML системе Business Notifications. На основании содержания документа XML и созданного события система Business Notifications отправит информацию подписчикам. Например, при возникновении события "Sales Order Created" система определит подписчиков данного события и отправит им уведомления по электронной почте о созданном заказе продажи.

Можно создавать более сложные события, чем простое "Sales Order Created". Например, можно указать, что событие должно создаваться, в том случае если итоговая сумма заказа продажи превысит определенную сумму.

События создаются в XMLport Event Designer, чтобы открыть его щелкните Вид, XMLport Events. На следующем рисунке показан XMLport Event Designer с тремя событиями для Purchase Order XMLport:

DRAFT



Пример

Нужно создавать событие с уведомлением при блокировке покупателя (значение поля Blocked изменено с False на True). Создается XMLport, который содержит необходимые данных, например No. и Name клиента, а также дату выполнения изменения. В XMLport Event Designer создается событие "Blocked". Наконец, нужно добавить следующий код в триггер OnModify таблицы ***Customer*** (CustXMLDoc это локальная переменная, ссылающая на XMLport):

```
OnModify()
IF Blocked = TRUE THEN
 IF Blocked <> xRec.Blocked THEN
  CustXMLDoc.Blocked();
```

Дальнейшую информацию см. в онлайновой справке к Business Notifications.

DRAFT

ЧАСТЬ 9 ОБЪЕКТЫ MENUSUITE

ГЛАВА 22 ОБЪЕКТЫ MENUSUITE

В данной главе содержится описание объекта MenuSuite, содержание которого отображается в Панели Навигации, также будет приведено описание Navigation Pane Designer.

Данная глава содержит информацию по следующим темам:

- MenuSuite – базовые понятия
- Настройка MenuSuite
- Экспорт объекта MenuSuite
- Обновление содержимого MenuSuite

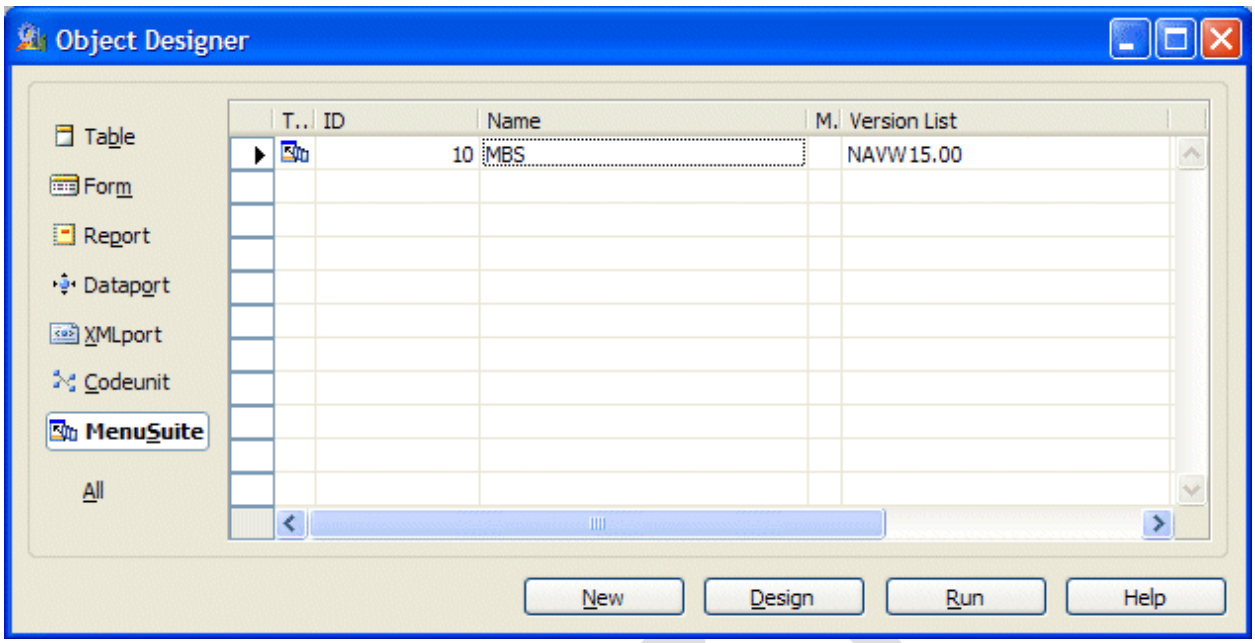
22.1 MenuSuite – Базовые Понятия

Объект MenuSuite содержит наборы меню, которые отображаются в Панели Навигации. Состав меню связан с определенной функциональной областью, например, Финансы или Производство.

Создание и Изменение Объектов MenuSuite

Чтобы создать объект MenuSuite или изменить существующий, щелкните кнопку **MenuSuite** в **Object Designer**, а затем щелкните кнопку **New** или **Design**.

DRAFT



Щелкнуть кнопку New. При этом откроется окно, в котором потребуется указать уровень создаваемого объекта. Если для всех уровней уже созданы объекты MenuSuite появится соответствующее системное сообщение. После того как будет выбран уровень, откроется Navigation Pane Designer. В заголовке Navigation Pane Designer будет показан уровень, над которым ведется работа.

Выбрать объекта MenuSuite и щелкнуть кнопку Design. При этом откроется Navigation Pane Designer, в котором будет отображаться содержимое выбранного объекта. В заголовке Navigation Pane Designer будет показан уровень, над которым ведется работа.

Кроме того доступны следующие возможности:

- Можно щелкнуть Файл, Import/Export, и Сервис, Translate, Import/Export.
- Можно компилировать объект MenuSuite (F11 или Сервис, Compile). Если в объекте MenuSuite содержатся ссылки на несуществующие формы, отчеты, пакетные задания, датапорты или программные модули, то возникнет ошибка компиляции.

Объект MenuSuite запустить нельзя.

22.2 Настройка MenuSuite

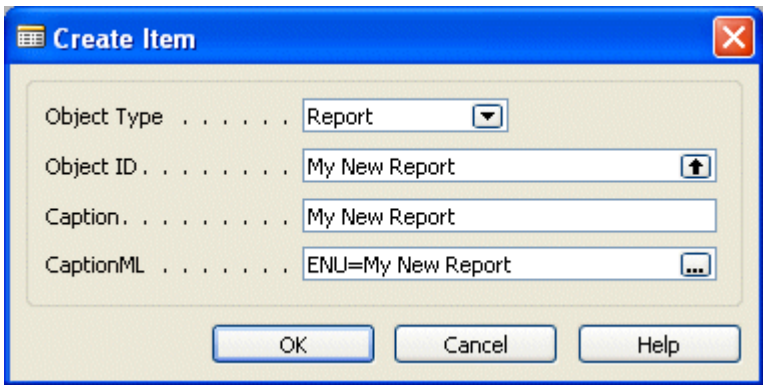
Настройка MenuSuite выполняется в Navigation Pane Designer. Далее будет написано о возможностях по настройке MenuSuite, доступных разработчикам. Доступ к этим возможностям осуществляется щелчком правой кнопкой мыши по кнопкам меню, группам меню и пунктам меню, а также в пустом месте панели навигации.

Смотрите онлайновую справку по Dynamics NAV для получения информации о возможностях предоставляемым администраторам, эти же возможности могут использоваться разработчиками.

Создание Пунктов Меню

Чтобы создать пункт меню, щелкните правой кнопкой мыши по пустому месту в панели навигации, или по группе меню, или по пункту меню и выберите Создать Пункт. Откроется окно ***Создать Пункт***, в котором нужно указать тип объекта, его идентификатор, а также подпись и captionML (многоязыковая подпись). Ниже показан пример:

DRAFT

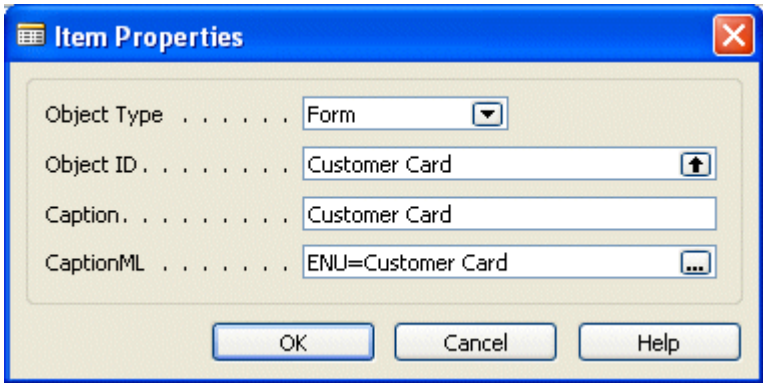


Например, вы создали новый отчет и хотите добавить его как пункт меню в какое-либо меню. В поле **Тип Объекта** в выпадающем списке выберите Report, в поле **Object ID** используйте кнопку Выбор, чтобы выбрать отчет из Report List. Когда вы щелкните по полю **Подпись**, оно автоматически будет заполнено автоматические. Вы можете изменить его при необходимости.

В поле **CaptionML** по умолчанию будет отображаться код выбранного языка и название пункта меню. Если пункт меню для разных пользователей должен отображаться на разных языках (в зависимости от используемого языка в системе), то нужно указать название пункта меню на всех этих языках. Если щелкнуть кнопку AssistButton в поле **CaptionML**, откроется окно ***Multilanguage Editor***, где можно ввести названия пункта меню для каждого языка:

Установка Свойств

Для каждого пункта меню есть четыре свойства, которые отображаются в окне ***Свойства Пункта***: *Тип Объекта*, *Объект ID*, *Подпись* и *CaptionML*. Чтобы открыть окно, щелкните по пункту меню правой кнопкой мыши и выберите Свойства.



Свойство	Описание
Тип Объекта	Значение в поле **Тип Объекта** можно выбрать из выпадающего списка. Доступны следующие типы объектов: Table, Form, Report, Codeunit, Dataport.
Объект ID	Используйте кнопку Выбор в поле **Объект ID,** чтобы выбрать требуемый объект из списка. Например, если в поле Тип Объекта указать Report, то для выбора откроется окно Report List.
Подпись	По умолчанию, поле **Подпись** содержит название, присвоенное пункту меню. При необходимости его можно изменить.
CaptionML	Поле **CaptionML** содержит код, используемого в системе языка и название пункта меню, например, ENU=General Journals. Если пункт меню используется на разных языках, то щелкните кнопку AssistButton в поле **CaptionML**, чтобы открыть окно ***Multilanguage Editor***, где можно ввести названия пункта меню для каждого языка

Уровень Объекта MenuSuite

Microsoft Business Solutions поставляет общий объект MenuSuite, который называется MBS. Перед тем как конечный пользователь увидит содержание данного объекта, он подвергается ряду изменений. Эти изменения применяются на различных уровнях, которые делятся на Country/Region, Add-on, Partner и Company. Например, изменения, выполняемые при локализации приложения, находятся на уровне Country/Region. Объект MenuSuite изменяемый партнером Microsoft Business Solutions, сохраняется на уровне Partner. Также можно изменить объект MenuSuite на уровне Company, на котором обычно работает администратор.

Изменения, выполняемые для объекта MenuSuite, хранятся как отличия между текущим уровнем MenuSuite и предыдущим. Например, когда администратор выполняет настройку MenuSuite на уровне Company, изменения хранятся как различия между уровнями Company и Partner. Если выгрузить объект MenuSuite в текстовый формат и открыть текстовый файл, то можно увидеть информацию о сделанных изменениях относительно предыдущего уровня.

Кроме уровня Add-on, на котором может существовать до 10 объектов MenuSuite, на каждом уровне разрешается только 1 объект MenuSuite.

22.3 Экспорт Объекта MenuSuite

Объект MenuSuite можно экспортировать в формате объектов Dynamics NAV (*.fob) или текстовом формате. Текстовый файл содержит изменения относительно предыдущего уровня. Далее будет показан пример текстового файла, содержащего изменения объекта MenuSuite на уровне Company.

Пример

Когда администратор выполнил настройку MenuSuite на уровне Company, изменения сохранятся на уровне Company. Когда объект MenuSuite для уровня Company будет выгружен в текстовый формат, текстовый файл будет содержать различия относительно объекта предыдущего уровня - Partner. Предположим, что администратор выполнил следующие изменения:

- В меню «Finance» удалил группу меню «Setup», в которой был один пункт меню «Accounting Periods».
- Добавил новый пункт меню «G/L Account Card» в группу меню «General Ledger» меню «Finance».
- Переместил пункт меню «G/L Account Card» так, чтобы он находился между пунктами меню «Chart of Accounts» и «Bank Accounts» группы меню «General Ledger.

Текстовый файл будет содержать следующую информацию:

```
OBJECT MenuSuite 70 Company
{
OBJECT-PROPERTIES
{
Date=19-04-04;
Time=16:04:10;
Modified=Yes;
Version List=;
}
PROPERTIES
{}
MENUNODES
{
{ MenuItem ;[{FA8395A4-7A0B-4524-B0FD-20436D9711A4}] ;Name=G/L Account Card;
CaptionML=ENU=G/L Account Card;
MemberOfMenu=[{F8D2429D-034B-4C58-9B5E-81BE962DB1BC}];
RunObjectType=Form;
RunObjectID=17;
ParentNodeID=[{B12180CF-0EFB-43AD-9118-7765E953AAFD}];
Visible=Yes;
NextNodeID=[{7A5B6DDE-B44B-41A0-95DA-69B7109F0E32}] }
{ ;[{1BB06483-AFFD-4750-BF62-3ABA035E11B7}]; Deleted=Yes}
{ ;[{DF601C8A-07F7-4841-8929-9F2065BCB302}]; Deleted=Yes}
{ ;[{8AC7917D-2C91-457D-80D6-A24B42F71AE7}];
NextNodeID=[{FA8395A4-7A0B-4524-B0FD-20436D9711A4}]}
}
}
```

Содержимое объекта a MenuSuite можно описать с помощью следующих характеристик:

- Состоит из набора меню.
- Меню содержит набор узлов, которые отображаются в Панели Навигации/Navigation Pane Designer в виде дерева.
- Узел меню может быть либо группой меню, либо пунктом меню.
- У узла меню есть глобальный уникальный идентификатор (GUID) и различные свойства.
- Группы меню содержат набор узлов.

DRAFT

- Пункт меню является самым низким уровнем в дереве. При щелчке по пункту меню запускаются связанные с ним формы, отчеты, пакетные задания, датапорты, программные модули.

22.4 Обновление Содержимого MenuSuite

Меню в MenuSuite, которые были унаследованы от объекта MenuSuite предыдущего уровня, отмечаются символом ">>", расположенного слева от названия меню. Эти меню будут обновлены, при обновлении соответствующих объектов MenuSuite. Любые изменения, выполненные для данных меню, будут объединены с новым MenuSuite, полученным в результате обновления. Например, если в группу меню был добавлен пункт меню, а содержимое группы меню в результате обновления изменилось, добавленный пункт будет помещен, как последний элемент группы меню.

Однако в некоторых случаях объединение не может быть выполнено. Например, если был добавлен пункт меню в группу, которая в результате обновления была удалена. Так как выполнить объединение невозможно, такие пункты меню будут помещены в группу Lost Items в самом низу дерева меню. Другим примером, может быть добавление пунктов и групп меню в меню, которое в результате обновления было удалено. В этом случае используется меню Lost Items.

После обновления можно решить, что делать с группой/меню Lost Items. Элементы можно сохранить в текущем меню (касается группы Lost Items), можно сохранить в другом меню или удалить их. При работе пользователя, ни группа, ни меню Lost Items в панели навигации не видны.

Новые меню, которые вы создали, при обновлении не изменяются. Если после обновления появились пункты меню, которые должны быть вставлены в новые меню, их стоит добавлять вручную.

DRAFT

ЧАСТЬ 10 МНОГОЯЗЫКОВАЯ ФУНКЦИОНАЛЬНОСТЬ

ГЛАВА 23 МНОГОЯЗЫКОВАЯ ФУНКЦИОНАЛЬНОСТЬ

В данной главе будут объяснены некоторые аспекты, касающиеся Многоязыковой Функциональности в Dynamics NAV.

В данной главе будут рассмотрены следующие темы:

- Многоязыковая функциональность
- Понимание базового языка, используемого при написании кода
- Диапазоны идентификаторов для текстовых констант

23.1 Многоязыковая Функциональность

Определение Текущего Языка Приложения

Чтобы определить текущий язык приложения среда C/SIDE выполняется функцию ApplicationLanguage (триггер с ID 4), в программном модуле 1. Этот триггер должен возвращать значение типа integer (language ID). Триггеру не разрешен доступ к базе данных. Если триггер не содержит код языка, среда C/SIDE считает его из файла fin.stx, который содержит общие текстовые строки, используемые C/SIDE.

Данный алгоритм был встроен в среду C/SIDE, чтобы обработать иерархию доступных языков. Данный алгоритм определяет, какой язык следует использовать, если одна и более текстовых строк пропущены для текущего языка приложения. Дополнительную информацию см. в разделе Отображение Текста. Дополнительную информацию об идентификаторах языков смотри в разделе Виртуальная Таблица «Windows Language».

Выбор Языка из Интерфейса Пользователя

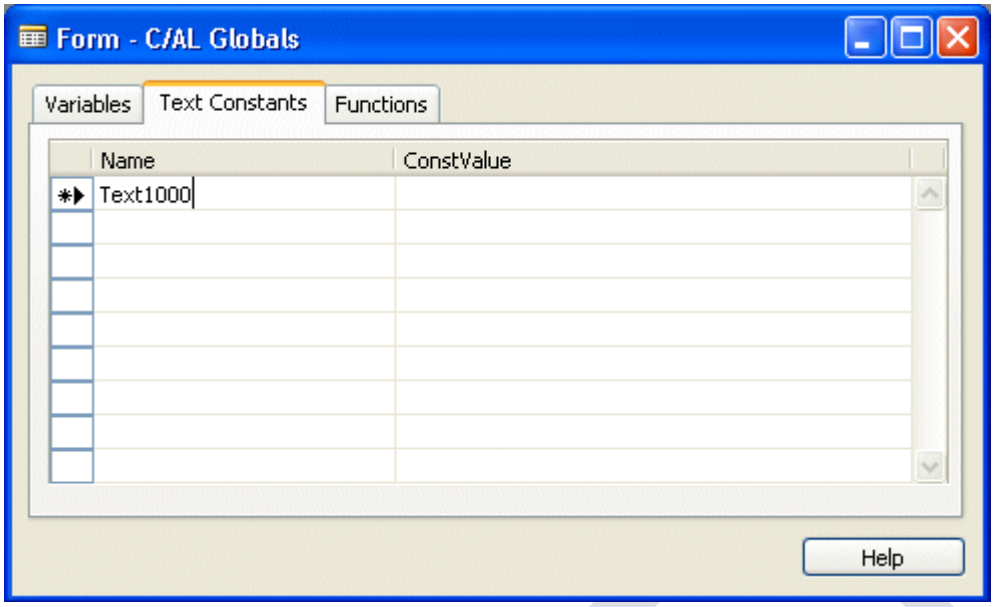
В базе данных, поддерживающей несколько языков, при щелчке пользователем по Сервис, Язык, триггер SetGlobalLanguage откроет форму 534. В окне ***Application Languages*** пользователь сможет выбрать язык, который будет использоваться при отображении заголовков окон, командных кнопок и т.п.

Текстовые Константы

В окнах ***C/AL Globals*** и ***C/AL Locals*** есть закладка ***Text Constants***. В данной закладке есть скрытый столбец **ConstValueML**, в котором отображаются все языки текстовой константы.

DRAFT

Текстовые константы применяются вместо жесткого программирования текстовых строк, зависящих от языка.



Дополнительную информацию о создании текстовых констант см. раздел Определение Переменных, Текстовых Констант и Функций в Программном Модуле

Языковые Модули

Языковой модуль содержит ту же информацию, что и файлы данных, полученные в результате использования Translate Import/Export. Однако языковой модуль, содержит текст только для одного языкового слоя. Языковые модули являются бинарными файлами, которые не могут быть изменены внешними инструментами.

Чтобы импортировать языковой модуль, щелкните Сервис, Языковой Модуль, Импорт. Для экспорта щелкните Сервис, Языковой Модуль, Экспорт.

Установка Файлов *.STX, *.ETX, *.CHM и *.HH для Многоязыковой Поддержки

Для каждого языка, к которому пользователь будет иметь доступ, нужно установить в соответствующие подкаталоги файлы *.stx, *.etx, *.chm и *.hh. Название подкаталога должно быть трехбуквенной аббревиатурой, используемой Windows для каждого языка. Дополнительную информацию об идентификаторах языков смотри в разделе «Виртуальная Таблица Windows Language».

Если создание подкаталога и установка в него файлов *.stx, *.etx, *.chm и *.hh осуществлялась при запущенном клиенте Dynamics NAV, чтобы новый языковой слой стал доступен нужно перезапустить программу.

Добавление Языкового Слоя

Чтобы позволить пользователю выбрать определенный язык из меню Сервис, этот язык должен присутствовать как гранула в файле лицензии.

Кроме того приложение должно быть переведено на данный язык, в базу данных перевод можно загрузить щелкнув Сервис, Translate, Import. Также можно выгрузить все текстовые строки и перевести их в специальном инструменте, таком как Dynamics NAV Localization Workbench, также перевод текстовых строк можно выполнить прямо в Multilanguage Editor.

Чтобы добавить перевод в Multilanguage Editor:

1. Открыть Object Designer и открыть объект, в который нужно добавить перевод.

DRAFT

2 Открыть окно ***Properties*** (SHIFT+F4).

3 Щелкнуть кнопку AssistButton в свойстве *CaptionML*, чтобы открыть окно ***Multilanguage Editor***.

4 Щелкнуть кнопку AssistButton в поле **Language**, чтобы открыть окно ***Windows Language List***.

5. Выбрать требуемый язык.

Можно просто ввести трехбуквенный код требуемого языка, а затем переместить курсор в поле **Value**. Система автоматически заменит аббревиатуру полным описанием языка.

6. В поле **Value** ввести корректное название данного объекта на выбранном языке.

7. Щелкнуть кнопку OK, чтобы сохранить введенную информацию.

Чтобы новый языковой слой мог работать с приложением, нужно в подкаталог для данного языка поместить подходящий файл fin.stx. Другими словами, чтобы пользователь мог выбирать определенный язык из меню Сервис, должно быть выполнено следующее:

- Должен существовать подкаталог с правильным наименованием.
- В подкаталоге должен находиться корректный файл fin.stx.
- Текстовые строки в базе данных, должны иметь перевод на соответствующий язык (language ID).
- Файл лицензии должен включать корректную гранулу.

Подкаталог для Языка

Каждый язык, к которому пользователь должен иметь доступ, должен быть представлен отдельным подкаталогом в каталоге Dynamics NAV.

В каждом подкаталоге должны быть:

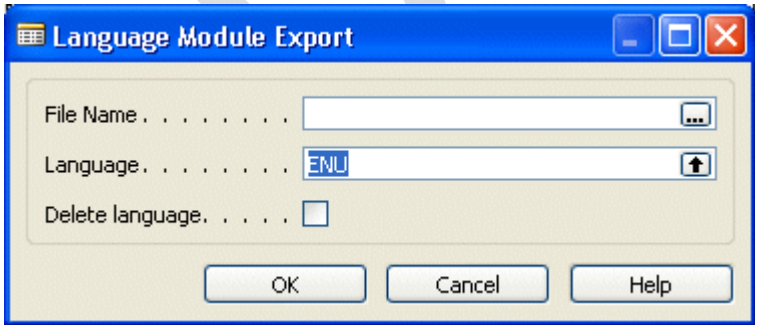
- Файл fin.stx
- Файл fin.etx
- Файлы онлайновой справки (*.chm и *.hh).

Удаление Языкового Слоя

После того как языковой слой был интегрирован в базу данных, существует только один способ удалить его.

Чтобы удалить языковой слой:

1. Щелкните Сервис, Языковой Модуль, Экспорт, чтобы открыть окно ***Language Module Export***:



2. В поле **Language** выберите язык, который больше не нужен (можно использовать для выбора окно ***Windows Language List)***.

3. В окне ***Language Module Export*** установите флаг в поле **Delete language** и щелкните кнопку OK.

DRAFT

Виртуальная Таблица «Windows Language»

Виртуальная таблица ***Windows Language***, это таблица только для чтения, в которой отображаются поддерживаемые языки Windows. Чтобы увидеть ее содержимое нужно создать табличную форму, основанную на таблице.

В виртуальной таблице ***Windows Language*** содержаться следующие поля:

Поле	Описание
Language ID	Данное поле является первичным ключом. В нем отображается стандартный идентификатор языка Windows (Windows Language ID). C/AL поддерживает настройку языка с помощью GLOBALLANGUAGE, WINDOWSLANGUAGE и свойств объекта LANGUAGE. Значения данных свойств берутся из данного поля.
Primary Language ID	Языки Windows сгруппированы. Группы состоят из первичного языка и ноля или нескольких вторичных языков. В поле **Primary Language ID** содержится идентификатор языка Windows (Windows Language ID) первичного языка
Name	Стандартное имя Windows для языка
Abbreviated Name	Это поле является вторичным ключом. В нем указывается стандартная трехбуквенная аббревиатура Windows для языка.
Enabled	Данный флаг показывает доступность языка. Файл лицензии определяет, как будет использоваться каждый язык.
Globally Enabled	Файл лицензии позволяет использовать данный язык как глобальный язык приложения.
Form Enabled	Файл лицензии позволяет формам отображаться на языке отличном от глобального языка.
Report Enabled	Файл лицензии позволяет отчетам печататься на языке отличном от глобального языка.
Dataport Enabled	Файл лицензии позволяет датапортам отображаться на языке отличном от глобального языка.
Primary Code Page	Кодовая страница для языка определяет набор символов, доступных для языка. Если смешать текст с использованием нескольких кодовых страниц, результат будет не предсказуем
STX File	Флаг в данном поле означает, что для данного языка установлен файл *.stx. Файл *.stx содержит текстовые значения, используемые средой C/SIDE, например, надписи меню, названия системных таблиц.
ETX File	Флаг в данном поле означает, что для данного языка установлен файл *.etx. Файл *.etx содержит сообщения об ошибках.
Help File	Флаг в данном поле означает, что для данного языка установлены файлы *.hlp или *.chm.

Закладки

Если создан элемент управления закладки, без указания значений в свойстве *PageNames*, среда C/SIDE будет использовать имена 0, 1, 2, и т.д. для страниц, содержащих видимые элементы управления. Станицы, которые не содержат видимые элементы управления, не отображаются.

DRAFT

Поддерживать Виды

В SQL Server Option для Dynamics NAV можно активизировать параметр Поддерживать Виды. Этот параметр определяет, будут ли на SQL-сервере создаваться и поддерживаться представления данных для каждого кода языка, добавляемого в поле или в таблицу Microsoft Dynamics NAV.

Если эта опция активирована, SQL-представления будут использоваться внешними программными средствами для получения доступа к свойству CaptionML объекта на соответствующих языках (а не к имени, содержащемуся в таблице).

NAV ODBC

В NAV ODBC включена поддержка нескольких языков. Пользователи NAV ODBC могут получать данные из Dynamics NAV на различных языках независимо языка приложения.

В NAV ODBC включает следующую многоязыковую функциональность:

- Названия таблиц
- Названия полей
- Значения OptionString
- Date Formula

Дополнительную информацию см. в руководстве *Microsoft Dynamics NAV ODBC Driver 5.0 Guide*.

Отображение Текста

Если среда C/SIDE должна отобразить текст, то она выполняет поиск этого текста на текущем языке. Если текст не найден, то выполняется поиск на другом языке.

Например, если пользователь использует язык German (Swiss) и хочет посмотреть форму, содержащую строку, которой не существует в German (Swiss), алгоритм указывает системе выполнить поиск этой строки на языке German (Standard). Причина заключается в том, что German (Standard) является первичным языком для German (Swiss).

Алгоритм, указывающий среде C/SIDE как выполнять поиск текста, использует следующий порядок:

1. Язык объекта
2. Первичный язык для языка объекта
3. Глобальный язык, выбранный пользователем
4. Первичный язык для глобального языка, выбранного пользователем
5. Язык приложения
6. Первичный язык для языка приложения

Несколько Языков в Документе

Если нужно, чтобы документы печатались на языке получателя, а не на выбранном языке приложения, нужно добавить всего одну строку в документ, для обработки такой ситуации. Эта функциональность уже включена для большинства печатных отчетов стандартной базы данных Dynamics NAV. Например, документ может быть напечатан на языке, указанном в поле **Language Code** окна ***Customer Card***.

В отчете, для которого требуется включить поддержку нескольких языков нужно вставить следующий код в первую строку триггера OnAfterGetRecord():

```
CurrReport.LANGUAGE := Language.GetLanguageID("Language Code")
```

DRAFT

Также для каждого отчета нужно создать новую переменную Language, с типом данных Record и таблицей ***Language*** (ID= 8).

После компиляции объекта, он будет печататься на языке указанном в окне ***Customer Card***.

23.2 Создание Приложений Поддерживающих Несколько Языков

При создании приложений поддерживающих несколько языков важно помнить о следующих трех правилах:

- Все свойства *Name* заполняются на английском языке (English (United States)).
- Вместо текстовых строк должны использоваться текстовые константы, например, в сообщениях об ошибках.
- У всего, что видит пользователь должно быть заполнено свойство *Caption*.

Перед началом работы в базе данных, поддерживающей несколько языков, нужно установить в качестве языка приложения английский. Чтобы сделать это щелкните Сервис, Язык и выберите English (United States).

Свойство Name

В Dynamics NAV код основывается на английском языке (English (United States)). Это означает, что свойство *Name* должно всегда быть на английском. Также сюда относятся:

- Названия объектов
- Названия полей
- Названия функций и переменных
- Комментарии
- Строки Option
- Названия элементов управления

Текстовые константы

Сообщения об ошибках и прочие текстовые строки должны вводиться как текстовые константы, в этом случае их легко переводить.

В базе данных поддерживающей один язык, сообщение об ошибке можно было вводить непосредственно в код в виде текста. В базе данных поддерживающей несколько языков, оно должно вводиться следующим образом:

```
IF FileName = '' THEN
  ERROR(Text000);
```

В данном примере Text000 является именем текстовой константы в данном объекте. Текстовые константы должны создаваться в окне ***C/AL Globals.*** Дополнительную информацию о создании текстовых констант см. раздел Определение Переменных, Текстовых Констант и Функций в Программном Модуле.

При добавлении текстовой константы в существующий объект, ее можно назвать в соответствии с вашими требованиями. Среда C/SIDE присвоит уникальный идентификатор согласно диапазонам, приведенным в разделе 23.4 Диапазоны Идентификаторов для Текстовых Констант, это позволит легко обновлять измененные объекты.

Если при работе с C/AL Editor поместить курсор на текстовую константу, содержимое текстовой константы будет показано в строке состояния на языке, который выбран в качестве языка приложения.

DRAFT

Свойство Caption

У всего, что видит пользователь должно быть заполнено свойство *Caption*. Если значение свойства *Name* всегда на английском (English (United States)), то свойство *Caption* используется для отображения пользователю названий на его языке.

Например, если нужно обратиться к полю из кода, в связи с сообщением об ошибке, для этого будет использовано свойство *Name*, но отображаться должно свойство the *Caption*:

```
VATPostingSetup.FIELDCAPTION("VAT Calculation Type")
```

где "VAT Calculation Type" это значение свойства *Name* на английском, а функция FIELDCAPTION гарантирует, что будет показано значение свойства *Caption* на соответствующем языке.

При программировании нужно всегда помнить разницу между свойствам *Name* и *Caption*, чтобы получать правильный результат при выполнении кода.

Свойство CaptionML

Свойство *CaptionML* позволяет изменять языки. Для всего должно существовать свойство *CaptionML* с корректным термином на английском (English (United States)). Кроме данного значения в свойстве могут указываться любые переводы объекта или поля. В свойство *Caption* копируется значение на языке приложения из поля *CaptionML*.

> **Пример**
>
> Таблица 37, у поля 1 следующие значения свойства CaptionML:
>
> ENU=Document Type;RUS=Тип документа

В свойстве **CaptionML** можно вводить значения как непосредственно в поле Value, так и щелкнув кнопку AssistButton, чтобы открыть окно Multilanguage Editor и вводить значения там.

Чтобы сохранить информацию, введенную в ***Multilanguage Editor*** нужно щелкнуть кнопку OK.

> **Примечание**
>
> Если создать новое поле на форме, содержимое свойства *Caption* не будет отображаться на форме, пока она не будет откомпилирована. Если вы скопировали другой элемент на форме и изменили его свойства, содержимое свойства *Caption* будет сразу отображаться в Form Designer. Тоже правило действует на формы запроса для отчетов.
>
> Если значение было введено непосредственно в свойство *Caption* (а не в CaptionML), то оно сразу будет отображаться на форме.

Создание Caption

Если в вашем приложении не переведены все значения свойства *Caption* это нужно сделать. Это можно сделать как вручную, так и с помощью инструмента make-ml. Дополнительную информацию об этом инструменте смотрите в руководстве *Upgrade Toolkit for Dynamics NAV*.

Особое внимание следует уделить:

- Кнопкам выбора
- Строкам Option
- Переменным Option

Кнопки Выбора (Option Buttons)

Нужно убедиться, что значение свойства *CaptionML* для кнопок выбора установлено корректно.

> **Примечание**

DRAFT

Значение в поле **OptionValue** всегда должно быть на английском потому, что значения используются в коде, а код должен быть на английском (English (United States)).

Также нужно убедиться, что значение для языка English (United States) в свойстве **CaptionML** совпадает со значением свойства **OptionValue**.

Строки Option

Нужно убедиться, что значение свойства *CaptionML* для строки option установлено корректно. Строка option может использоваться в элементе управления, например, на форме запроса.

Переменные Типа Option Variables

При использовании переменной типа option, например в выражениях: FORMAT, STRSUBSTNO, ERROR, MESSAGE, или CONFIRM, нужно использовать SELECTSTR с текстовой константой, содержащей строку option.

Date Formula

При создании в таблице поля, которое должно содержать формулу даты, для данного поля следует использовать тип данных DateFormula. Этот тип данных не зависит от языка и предоставляет функции CALCDATE возможности по работе с несколькими языками.

Можно достичь похожего же результата используя для поля тип данных Code или Text, и установив в свойстве *Dateformula* значение Да. Однако это решение зависит от языка, что означает для пользователей использующих разные языки отсутствие возможности использовать одни данные.

Использование в C/AL Коде

При использовании функции CALCDATE для расчета дат, нужно вводить формулу даты на английском, но при этом использовать угловые скобки (< >) вокруг формулы даты. Теперь формула даты будет работать корректно независимо от языка приложения, результаты вычисления всегда будут одинаковыми, не смотря на язык, выбранный пользователем.

Пример

```
EndOfMonth := CALCDATE('<CM>',TODAY)
```

23.3 Понимание Базового Языка, Используемого при Написании Кода

Если вы не привыкли использовать английский в качестве базового языка, среда C/SIDE поможет вам привыкнуть к работе в новом окружении. Для этого в среде C/SIDE есть несколько механизмов.

Как Увидеть Одновременно и Caption и Name

В различных окнах среды C/SIDE можно одновременно увидеть значения свойств *Name* и *Caption* для выбранного элемента.

Функциональность «Раскрыть»

При использовании функциональности «Раскрыть» на объекте, можно указать системе, что требуется отображать поля на локальном языке и на English (United States).

1. Откройте объект, например, заказ покупки. Поместите курсор в подходящее поле.

2. Щелкните Сервис, Раскрыть (CTRL+F8).

3. В окне ***Все Поля***, щелкнуть Вид, Показать Столбец.

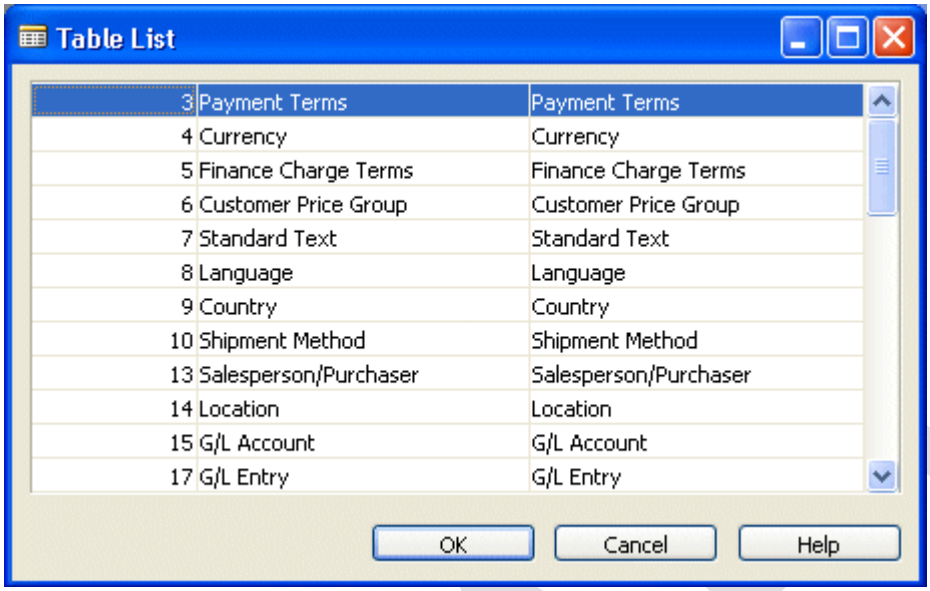
4. В окне ***Показать Столбец*** установить флаг напротив **Имя Поля** и щелкнуть OK.

DRAFT

В столбце ***Поле*** теперь отображается Caption на выбранном языке приложения, а в столбце ***Имя Поля*** отображается свойство Name.

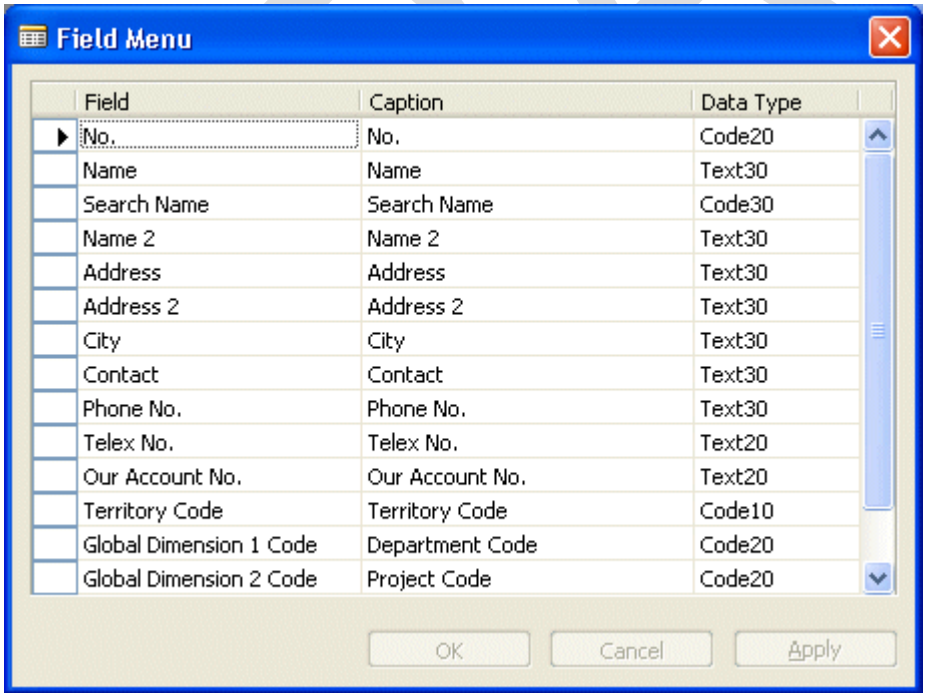
Table List, Form List, Field List, Object List и Field Menu

В окнах ***Object Designer***, ***Table List***, ***Form List***, ***Field List***, ***Object List*** и ***Field Menu*** можно видеть и значение свойства *Name*, и значение свойства *Caption*:



Первый столбец слева содержит номер объекта, во втором значение свойства *Name*, а в третьем содержится значение свойства *Caption* на текущем языке приложения.

В окнах ***Object Designer*** и ***Field Menu*** можно скрыть столбец **Caption**, щелкнув по заголовку столбца **Caption** правой кнопкой мыши и выбрав пункт Скрыть Столбец.



Столбец можно снова отобразить, для этого нужно выполните такие же действия как описано в разделе «Функциональность «Раскрыть».

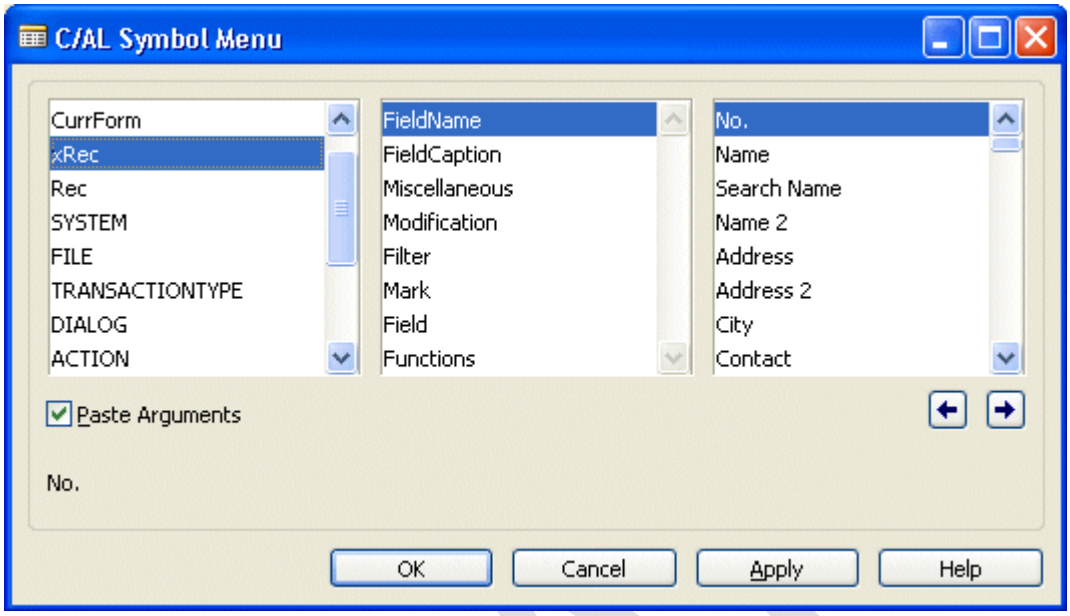
C/AL Сканер

В C/AL Editor сканер показывает значения свойства caption для объектов, полей и текстовых констант. Так как код должен быть на английском, сканер помогает правильно его понимать.

Если поместить курсор на объект, поле или текстовую константу в C/AL коде, сканер найдет для него значение свойства caption на текущем языке приложения и отобразит найденное значение в строке состояния (в нижней части окна Dynamics NAV).

C/AL Symbol Menu

В окно ***C/AL Symbol Menu*** была добавлена отдельная категория: *FieldCaption*. Ее можно видеть при выборе переменной, связанной с таблицей:



Если выбрать название поля, то в нижнем углу окна будет отображен заголовок для данного поля на текущем языке приложения. Другими словами, в третьем столбце ***C/AL Symbol Menu*** отображается значение свойства *Name*, а в нижнем углу окна будет отображаться значение свойства *Caption*. На рисунке текущий язык приложения английский, поэтому у обоих свойств одинаковые значения. Если выбрать заголовок поля, то в нижнем углу окна будет отображено название для данного поля на текущем языке приложения. Другими словами, в третьем столбце ***C/AL Symbol Menu*** отображается значение свойства *Caption*, а в нижнем углу окна будет отображаться значение свойства *Name*.

23.4 Диапазоны Идентификаторов для Текстовых Констант

C/SIDE присваивает уникальные идентификаторы текстовым константам согласно следующей таблице диапазонов идентификаторов:

Разработчик	От	До
Microsoft Business Solutions HQ	000	9,999
Microsoft Business Solutions Netherlands	1,000,000	1,009,999
Microsoft Business Solutions Belgium	1,010,000	1,019,999
Microsoft Business Solutions USA	1,020,000	1,029,999
Microsoft Business Solutions Canada	1,030,000	1,039,999
Microsoft Business Solutions United Kingdom	1,040,000	1,049,999
Microsoft Business Solutions Iceland	1,050,000	1,059,999
Microsoft Business Solutions Denmark	1,060,000	1,069,999
Microsoft Business Solutions Sweden	1,070,000	1,079,999

Разработчик	От	До
Microsoft Business Solutions Norway	1,080,000	1,089,999
Microsoft Business Solutions Finland	1,090,000	1,099,999
Microsoft Business Solutions Spain	1,100,000	1,109,999
Microsoft Business Solutions Portugal	1,110,000	1,119,999
Microsoft Business Solutions France	1,120,000	1,129,999
Microsoft Business Solutions Italy	1,130,000	1,139,999
Microsoft Business Solutions Germany	1,140,000	1,149,999
Microsoft Business Solutions Switzerland	1,150,000	1,159,999
Microsoft Business Solutions Austria	1,160,000	1,169,999
Microsoft Business Solutions Poland	1,170,000	1,179,999
Microsoft Business Solutions Lithuania	1,180,000	1,189,999
Microsoft Business Solutions Latvia	1,190,000	1,199,999
Microsoft Business Solutions Estonia	1,200,000	1,209,999
Microsoft Business Solutions Russia	1,210,000	1,219,999
Microsoft Business Solutions Czech Republic	1,220,000	1,229,999
Microsoft Business Solutions Slovenia	1,230,000	1,239,999
Microsoft Business Solutions Australia	1,240,000	1,249,999
Microsoft Business Solutions New Zealand	1,250,000	1,259,999
Microsoft Business Solutions Singapore	1,260,000	1,269,999
Microsoft Business Solutions South Africa	1,270,000	1,279,999
Microsoft Business Solutions India	1,280,000	1,289,999
Microsoft Business Solutions Argentina	1,290,000	1,299,999
Microsoft Business Solutions Brazil	1,300,000	1,309,999
Microsoft Business Solutions Mexico	1,310,000	1,319,999
Microsoft Business Solutions Croatia	1,320,000	1,329,999
Microsoft Business Solutions North Africa/Middle East	1,330,000	1,339,999
Microsoft Business Solutions Thailand	1,340,000	1,349,999
Microsoft Business Solutions Malaysia	1,350,000	1,359,999
Microsoft Business Solutions Hungary	1,360,000	1,369,999
Microsoft Business Solutions Ireland	1,370,000	1,379,999
General customer modifications	1,000,000,000	1,000,999,999
Add-on	1,100,000,000	1,199,999,999

DRAFT

ЧАСТЬ 11 УГЛУБЛЕННОЕ ИЗУЧЕНИЕ

ГЛАВА 24 SUMINDEXFIELDS

В данной главе будут рассмотрены SumIndexFields, которые являются основой для FlowFields в приложениях C/SIDE. Данная глава описывает, как технология SIFT работает на C/SIDE Database Server. Эта информация поможет программистам разрабатывать эффективные приложения, которые используют технологию SumIndexField.

24.1 SumIndexFields

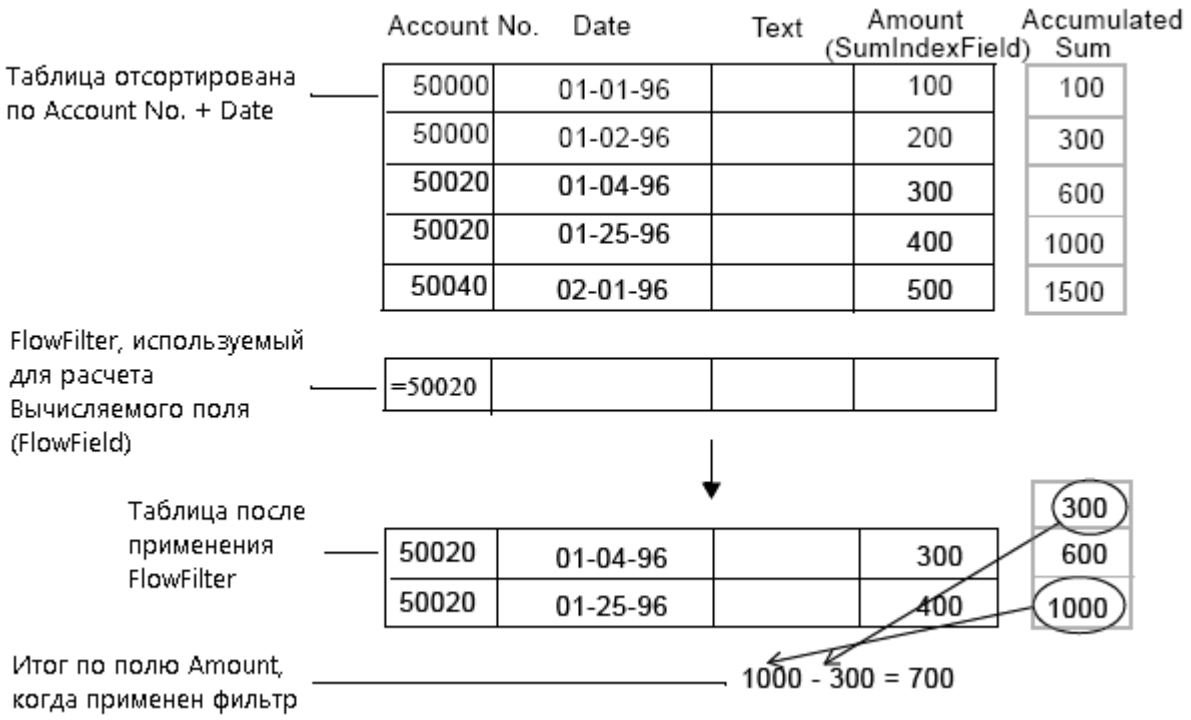
Технология SumIndexField (SIFT) была создана для улучшения производительности при выполнении таких операций как расчет баланса по клиенту. В традиционных системах баз данных предполагает необходимость многократного получения доступа и производства множества вычислений, прежде чем будет получен результат. Мощь и эффективность технологии SIFT в C/SIDE Database Server позволяет очень быстро получать суммы числовых столбцов, даже в таблицах содержащих тысячи записей. Эта функция используется во всем приложении Dynamics NAV и также поддерживается в SQL Server Option.

SIFT и C/SIDE Database Server

SumIndexField - фундаментальная особенность баз данных в формате Dynamics NAV, составляющая основу для FlowFields. Поля SumIndexField связаны с ключом, каждый ключ может включать максимум 20 SumIndexField. При проектировании базы данных, поле типа decimal может быть связано с ключом как SumIndexField. Это указывает СУБД, что необходимо создать и поддерживать структуру, в которой содержатся суммы значений в столбце. При выборе нового текущего ключа, все связанные с ним поля SumIndexField становятся доступными.

Следующий рисунок показывает таблицу, в которой поле **Amount** (столбец) определено как SumIndexField в ключе (Account No + Date). Это позволяет СУБД автоматически поддерживать итоговые суммы по столбцу. Каждый раз при изменении поля в столбце, автоматически обновляются итоговые суммы.

DRAFT



Справа от таблицы показана область базы данных, где хранятся итоговые суммы для столбца **Amount**. Третье поле в столбце с итоговыми суммами содержит значение 600, т.к. первыми тремя значениями поля **Amount** являются 100, 200 и 300, что в итоге дает 600. В четвертом виртуальном поле содержится значение 1000, итог по первым четырем значениям столбца **Amount**, и т.д. Если таблица содержит еще один SumIndexField, его значению будут суммироваться аналогичным образом.

Какие преимущества предоставляет SumIndexFields? Быстрый расчет сумм (по столбцам) и отображение результата в Вычисляемых Полях (FlowFields). Предположим, что нам нужна сумма всех значений в поле Amount. В традиционных системах, СУБД должна будет обратиться к каждой записи и сложить все значения поля Amount. Для базы данных с тысячами записей, это довольно времяемкая операция. В данном же случае нужно создать поле FlowField и указать в нем формулу расчета, как сумму по полю Amount. В результате СУБД должна будет просто получить значение из поля SumIndexField.

Операции с полями SumIndexFields остаются быстрыми при применении FlowFilter. Во второй таблице предыдущего рисунка показан набор записей, выбранных на основании фильтра FlowFilter по полю **Account No.**. Две записи удовлетворяют условиям фильтра. Чтобы получить итог по полю Amount нужно выполнить всего два обращения: одно чтобы получить итоговую сумму записи, предшествующей первой записи в выборке, а второй, чтобы получить итоговую сумму для последней записи в выборке.

Значение 300 было рассчитано на основании значения 1000, из которого вычли сумму коррекции (700). Не имеет значение количество записей в выбранном диапазоне, для расчета итога системе потребуется выполнить только два обращения.

SIFT созданы на основании индексов, используемых в C/SIDE Database Server и чем больше полей SumIndexFields добавляется к ключу, тем больше становится индекс. Однако благодаря специальной структуре индексов используемой в СУБД, время, затрачиваемое на поддержку итоговых сумм для полей SumIndexFields, является незначительным.

DRAFT

ГЛАВА 25 ПРЕОБРАЗОВАНИЕ ТИПОВ ДАННЫХ

В данной главе описаны все возможные преобразования типов данных в выражениях языка C/AL. Глава содержит следующие разделы:

- Преобразование Типов Данные в Выражениях
- Механизмы Преобразования Типов Данных

25.1 Преобразование Типов Данных в Выражениях

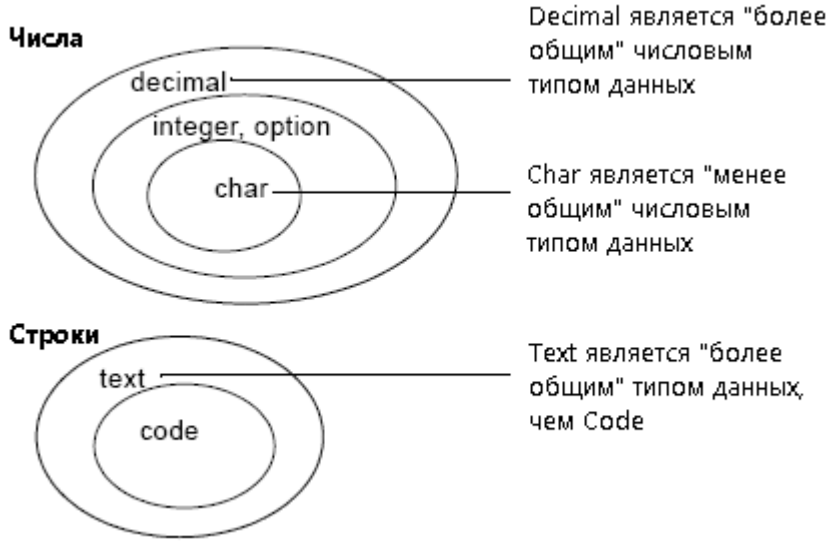
Рассмотрим следующий код на языке C/AL:

```
CharVar := 15; // переменная типа char
integerVar := 56000; // переменная типа integer
Sum := CharVar + integerVar;
```

В последней строке будет выполнено одно или два преобразования типов данных. В правой части, будет выполнен расчет выражения CharVar + integerVar (char +integer). Чтобы выполнить расчет выражения, первый операнд (CharVar) будет преобразован из типа данных char в integer. Оператор сложения возвратит результат в виде integer. Но если переменная в левой части была объявлена как decimal, результат должен быть преобразован из типа данных integer в decimal, перед тем как значение сможет быть присвоено переменной Sum (этот вид преобразования рассматривался в разделе «Присваивание и Преобразование Типов»)

В данной главе будут описаны преобразования типов данных, которые происходят при вычислении выражений. Вначале рассмотрим общие правила:

- При необходимости расчета выражения содержащего смешанный тип данных, система при возможности преобразует как минимум один операнд в более общий тип данных.
- Типы данных двух основных групп: числа и строки, могут быть проранжированы от «менее общий» к «более общий»



- Более общий тип данных включает все возможные значения менее общего типа данных: тип данных decimal более общий, чем integer, который в свою очередь более общий, чем char.
- Преобразование типов данных может выполняться даже в том случае, если оба операнда принадлежат одному типу данных.

Данные правила можно проиллюстрировать несколькими примерами.

Пример 1

Вычисление числового выражения:

integer + decimal

Данное выражение содержит два операнда, принадлежащих к различным типам данных. Перед тем как будет выполнена операция сложения, система должна преобразовать левый операнд в тип decimal:

decimal + decimal

После того как левый операнд был преобразован, можно вычислить выражение. Результатом вычисления будет тип данных decimal:

decimal + decimal = decimal

Пример 2

Вычисление строкового выражения:

text + code

Данное выражение содержит два операнда, для которых должны быть выполнена операция конкатенации. Чтобы сделать это, система должны преобразовать операнд менее общего типа данных (code) к более общему типу данных (text).

text + text

После того как правый операнд был преобразован, можно вычислить выражение. Результатом вычисления будет тип данных text.

text + text = text

25.2 Механизмы Преобразования Типов Данных

В данном разделе будут глубже рассмотрены механизмы преобразования типов данных для операторов C/AL. Для начала разделим операторы на несколько основных категорий:

- Операторы Сравнения (Relational operators)
- Логические Операторы (Logical operators)
- Арифметические Операторы (Arithmetic operators)

В следующих подразделах будут рассмотрены операторы C/AL каждой категории. В каждой категории операторов будет приведено описание корректных типов данных для аргументов, а также тип данных результата вычисления выражения.

Операторы сравнения будут рассмотрены первыми, так как они являются наиболее общими для большинства типов данных C/AL.

Операторы Сравнения (Relational Operators)

Операторы сравнения используются для сравнения выражений. В следующей таблице приведены правила вычисления операторов сравнения. Подразумевается, что типы данных можно сравнивать, см. Корректное Использование Операторов Сравнения, чтобы получить полный обзор сопоставимых типов данных.

Оператор	Название Оператора	Выражение	Тип данных результата
>	Больше чем	Expr > Expr	boolean
<	Меньше чем	Expr < Expr	boolean
<=	Меньше чем или равно	Expr >= Expr	Boolean

DRAFT

Оператор	Название Оператора	Выражение	Тип данных результата
<>	Не равно	Expr <> Expr	boolean
=	Равно	Expr = Expr	Boolean
IN	В диапазоне	Expr IN [Набор значений]	Boolean

Примечание

При использовании операторов сравнения, верхний и нижний регистр в строках имеет значение. Сравнение выполняется на встроенной таблице сравнения символов, которая не совпадает с таблицей символов ASCII.

Корректное Использование Операторов Сравнения

В следующей таблице показано корректное использование операторов сравнения, а также тип данных результата вычисления выражения. Недопустимые комбинации типов данных при использовании операторов сравнения отмечены черточкой. Все операторы сравнения являются бинарными операторами, это означает, что у них есть левый и правый операнд, а сам оператор помещается между ними.

Строки таблицы представляют левый операнд, а столбцы – правый. В ячейках показан тип данных результата.

На основании таблицы видно, что операторы сравнения можно использовать для сравнения значений типа text со значениями типа text или code, а значения типа boolean можно сравнивать только с другим значением типа boolean, и т.д.

Операторы Сравнения	bool	Char	option	integer	decimal	Date	time	text	code
Bool	Bool	-	-	-	-	-	-	-	-
Char	-	Bool	Bool	Bool	Bool	-	-	-	-
Option	-	Bool	Bool	Bool	Bool	-	-	-	-
Integer	-	Bool	Bool	Bool	Bool	-	-	-	-
Decimal	-	Bool	Bool	Bool	Bool	-	-	-	-
Date	-	-	-	-	-	Bool	-	-	-
Time	-	-	-	-	-	-	Bool	-	-
Text	-	-	-	-	-	-	-	Bool	Bool
Code	-	-	-	-	-	-	-	Bool	Bool

Логические Операторы (Boolean Operators)

Логические операторы могут использоваться только такими операндами, которые могут быть преобразованы в тип данных boolean.

Оператор	Название Оператора	Выражение	Тип данных результата

DRAFT

Оператор	Название Оператора	Выражение	Тип данных результата
NOT	Логическое отрицание	NOT bool	bool
AND	Логическое И	bool AND bool	bool
OR	Логическое ИЛИ	bool OR bool	bool
XOR	Исключающее ИЛИ	bool XOR bool	bool

Как видно в таблице, оператор NOT является унарным оператором. Это означает, что для него требуется один операнд, а сам оператор устанавливается перед операндом. Операторы AND, OR и XOR являются бинарными операторами, это означает, что у них есть левый и правый операнд, а сам оператор помещается между ними.

Арифметические Операторы (Arithmetic Operators)

Ниже приведено несколько примеров демонстрирующих правила преобразования типов данных для арифметических операторов. В примерах показано, как предполагается использовать операторы, а также показан эффект от выполнения преобразования типов данных, автоматически выполненного компилятором C/AL. Примеры разделены на группы в соответствии с типами данных языка C/AL.

Полное описание правил преобразования типов приведено в таблицах в разделе «Полный Обзор Правил Преобразования Типов Данных».

Пример

В данной таблице показано преобразование типов данных, при использовании в выражении операндов типа integer

Оператор	Название Оператора	Выражение	Тип данных результата
+	Унарный плюс	+ integer	integer
-	Унарный минус	- integer	integer
+	Сложение	integer + integer	integer
-	Вычитание	integer - integer	integer
*	Умножение	integer * integer	integer
/	Деление	integer / integer	decimal
DIV	Целочисленное деление	integer DIV integer	integer
MOD	Остаток от деления	integer MOD integer	integer

Такие же правила применяются для типа данных option.

Пример

В данной таблице показано преобразование типов данных, при использовании в выражении операндов типа decimal

DRAFT

Оператор	Название Оператора	Выражение	Тип данных результата
+	Унарный плюс	+ decimal	decimal
-	Унарный минус	- decimal	decimal
+	Сложение	decimal + decimal	decimal
-	Вычитание	decimal - decimal	decimal
*	Умножение	decimal * decimal	decimal
/	Деление	decimal / decimal	decimal
DIV	Целочисленное деление	decimal DIV decimal	decimal
MOD	Остаток от деления	decimal MOD decimal	decimal

Пример

В данной таблице показано преобразование типов данных, при использовании в выражении операндов типа date

Оператор	Название Оператора	Выражение	Тип данных результата
+	Увеличение даты	date + Number	date
-	Уменьшение даты	date - Number	date
-	Разница между датами	date - date	integer

В примерах «Увеличение даты» и «Уменьшение даты» возникнет ошибка выполнения, если операнд date является закрывающей датой или неопределенной датой (0D).

Пример

В данной таблице показано преобразование типов данных, при использовании в выражении операндов типа time

Оператор	Название Оператора	Выражение	Тип данных результата
+	Увеличение времени	time + integer	Time
-	Разница между временем	time - time	Integer

Единица измерения времени – миллисекунда. Если время неопределенно (0T), возникнет ошибка выполнения.

Пример

В данной таблице показано преобразование типов данных, при использовании в выражении операндов типа text и code (Строки)

Оператор	Название Оператора	Выражение	Тип данных результата

DRAFT

Оператор	Название Оператора	Выражение	Тип данных результата
+	Конкатенация	text + text	Text
+	Конкатенация	text + code	Text
+	Конкатенация	code + text	Text
+	Конкатенация	code + code	Code

Полный Обзор Правил Преобразования Типов Данных

В следующих таблицах приведено полное описание правил преобразования типов данных для арифметических операторов.

Унарные Арифметические Операторы

В языке C/AL для унарных операторов, также называемых префиксными операторами, используется следующий синтаксис:

PrefixExpression = PrefixOperator Expression

В таблице показаны типы данных, с которыми можно использовать унарные операторы, а также результирующие типы данных.

Унарный оператор	option	integer	decimal
+	Integer	Integer	decimal
-	Integer	Integer	decimal

Бинарные Арифметические Операторы

В данной таблице показаны типы данных, для которых применяются арифметические операторы.

Для всех бинарных арифметических операторов в C/AL используется следующий синтаксис:

InfixExpression = LeftExpression InfixOperator RightExpression

Operator	boolean	char	option	integer	decimal	date	time	text	code
+	○	●	●	●	●	●	●	●	●
-	○	●	●	●	●	●	●	○	○
*****	○	●	●	●	●	○	○	○	○
/	○	●	●	●	●	○	○	○	○
DIV	○	●	●	●	●	○	○	○	○
MOV	○	●	●	●	●	○	○	○	○

- ● Да, как минимум один из операндов (левый, правый или оба) может быть указанного типа.
- ○ Нет, данный тип данных не может использоваться с указанным оператором.

В следующих таблицах показано корректное использование бинарных арифметических операторов, а также тип данных результата вычисления выражения.

DRAFT

Правила Преобразования Типов Данных для Оператора «+»

Оператор «+»	boolean	char	Option	integer	decimal	date	Time	text	code
boolean	-	-	-	-	-	-	-	-	-
char	-	integer	integer (C)	integer (C)	integer (C)	-	-	-	-
option	-	integer (C)	integer (C)	integer (C)	integer (C)	-	-	-	-
integer	-	integer (C)	integer (C)	integer (C)	integer (C)	-	-	-	-
decimal	-	integer (C)	integer (C)	integer (C)	integer (C)	-	-	-	-
date	-	date(A) (C)	date(A) (C)	date(A) (C)	date(A) (C) (D)	-	-	-	-
time	-	time(B) (C)	time(B) (C)	time(B) (C)	time(B) (C) (D)	-	-	-	-
text	-	-	-	-	-	-	-	text	text
code	-	-	-	-	-	-	-	text	code

(A) нельзя использовать для даты 0D.

(B) нельзя использовать для времени 0T.

(C) может произойти перевыполнение.

(D) нельзя использовать, если у decimal есть дробная часть.

Правила Преобразования Типов Данных для Оператора «-»

Оператор «-»	boolean	char	option	integer	Decimal	date	Time	text, code
boolean	-	-	-	-	-	-	-	-
char	-	integer	integer(C)	integer(C)	integer(C)	-	-	-
option	-	integer(C)	integer	integer	decimal(C)	-	-	-
integer	-	integer(C)	integer	integer	decimal(C)	-	-	-
decimal	-	decimal(C)	decimal(C)	decimal(C)	decimal(C)	-	-	-
date	-	date(A)(C)	date(A)(C)	date(A)(C)	date(A)(C)(D)	integer(A)	-	-
time	-	time(B)(C)	time(B)(C)	time(B)(C)	time(B)(C)(D)	-	integer(B)	-
text	-	-	-	-	-	-	-	-
code	-	-	-	-	-	-	-	-

(A) нельзя использовать для даты 0D.

DRAFT

(B) нельзя использовать для времени 0T.

(C) может произойти перевыполнение.

(D) нельзя использовать, если у decimal есть дробная часть.

Правила Преобразования Типов Данных для Оператора «*»

Оператор «*»	char	Option	Integer	Decimal
Char	integer (C)	integer (C)	integer (C)	decimal (C)
option	integer (C)	integer (C)	integer (C)	decimal (C)
integer	integer (C)	integer (C)	integer (C)	decimal (C)
decimal	decimal (C)	decimal (C)	decimal (C)	decimal (C)

(C) может произойти перевыполнение.

Правила Преобразования Типов Данных для Оператора «/»

Оператор «/»	char	Option	Integer	Decimal
Char	decimal	decimal	decimal	decimal
Option	decimal	decimal	decimal	decimal
Integer	decimal	decimal	decimal	decimal
Decimal	decimal	decimal	decimal	decimal

Во всех случаях может произойти перевыполнение.

Произойдет ошибка выполнения, если правый операнд равен 0 (ноль).

Правила Преобразования Типов Данных для Операторов «MOD» и «DIV»

Операторы MOD и DIV	char	option	Integer	Decimal
Char	integer	integer	integer	decimal
option	integer	integer	integer	decimal
integer	integer	integer	integer	decimal
decimal	decimal	decimal	decimal	decimal

Произойдет ошибка выполнения, если правый операнд равен 0 (ноль).

DRAFT

ГЛАВА 26 НУМЕРАЦИЯ В DYNAMICS NAV

В данной главе будет объяснено, как в Dynamics NAV выполняется нумерация элементов, например документов. Информация будет полезна, если планируется использовать SQL Server Option для Dynamics NAV. Также рекомендуется ознакомиться с данной главой, если используется C/SIDE Database Server и требуется, чтобы Dynamics NAV выполнял сортировку данных таким же образом, как они выглядят во внешних приложениях.

Данная глава включает раздел:

- Как работает сортировка номеров?

26.1 Как Работает Сортировка Номеров?

Поля типа Code в Dynamics NAV могут содержать как числовые значения, так и текстовые строки. Dynamics NAV обеспечивает в C/SIDE Database Server корректную сортировку числовых значений, хранящихся в полях типа Code. Однако при использовании внешних приложений для доступа к тем же данным, сортировка может отличаться. Внешние программы могут читать и сортировать данные числа как текст. Числа, хранящиеся в полях code на SQL Server с использованием типа данных SQL *Varchar,* не будут отсортированы в корректном числовом порядке. SQL Server Option для Dynamics NAV сортирует числовые значения, словно это текстовые строки. В следующей таблице проиллюстрированы возможные различия:

Числовая Сортировка	Текстовая Сортировка
1	1
2	10
3	100
4	2
10	3
100	4

Чтобы избежать данной проблемы, рекомендуется использовать серии номеров фиксированной длины. Это можно сделать тремя способами:

- Создать серию номеров, которая состоит из предопределенного количества разрядов и начинается с цифры, отличной от 0 (ноль). Например, 100-399 (300 значений). Если данная серия номеров слишком короткая, то можно использовать больше знаков, например, 40,000-69,999 (30,000 значений) или 7,000,000-9,999,999 (3,000,000 значений). Пользователи быстро привыкают вводить числа фиксированной длины, а числа будут сортироваться правильно.

  1001

  1002

  1003

  ..

  ..

  9999

  Рекомендуется использовать это решение потому, что независимо от используемого типа данных SQL: *Varchar* или *Integer* сортировка будет выполняться корректно.

DRAFT

- Создать серию номеров, которая состоит из предопределенного количества разрядов и начинается с буквы. Например, A001-A999. Данная серия номеров будет сортироваться правильно. Когда серия номеров закончится, можно создать новую серию, которая начинается с другой буквы. Пользователи быстро привыкают вводить числа фиксированной длины, а числа будут сортироваться правильно.

  A001

  A002

  A003

  ..

  ..

  A999

- Создать серию номеров, которая состоит из предопределенного количества разрядов и начинается с ноля. Например, 001-999. Данное решение использовать не рекомендуется, так как оно содержит ряд недостатков. Во-первых, пользователи постоянно пытаются опустить ведущие нули и ввести непосредственно значение. Кроме того, SQL Server Option для Dynamics NAV не позволит сохранить номера, если для поля используется тип данных SQL *Integer*.

  Важно

  Главное правило, заключается в том, что типы данных для связанных полей должны быть совместимыми. Т.е. при изменении типа данных SQL для поля, также нужно изменить типы данных SQL связанных полей в других таблица. Например, если в таблице изменить тип данных SQL поля **No.** таблицы ***G/L Account*** с *Varchar* на *Integer* (или изменить тип данных с *Code* на *Text*), также нужно изменить типы данных в полях **G/L Account No.** таблиц ***G/L Entry*** и ***G/L Budget Entry***. Иначе, в плане счетов (таблица G/L Account) могут отображаться некорректные итоги, которые рассчитываются на указанных таблицах.

Принципы Нумерации

Чтобы обеспечить правильную сортировку чисел в полях типа code или text, независимо от варианта используемого сервера, следует использовать следующие принципы:

- *Всегда* использовать серии номеров фиксированной длины, например, 100-399.
- *Никогда* не использовать в полях типа code или text такие серии номеров как 1-999.
- *Никогда* не использовать в полях типа code или text такие серии номеров как 001-999.

Фильтры

Если вы не будете следовать принципам нумерации, то возможно возникновение проблем при наложении фильтров в Dynamics NAV.

Примеры:

- Если не использовать серию номеров фиксированной длины, то при наложении фильтра 10..20, результатом может стать 10,100.....20.
- Если не следовать принципам нумерации, то при наложении фильтра 2..10, не будет выбрано ни одной записи, т.к. 2 идет после 10.
- Если вы следуете принципам нумерации и используете серии номеров из трех разрядов и установите фильтр 10..20, результатом будет 100,101,102.....199.

DRAFT

ГЛАВА 27 СРЕДА C/SIDE В МНОГОПОЛЬЗОВАТЕЛЬСКОМ ОКРУЖЕНИИ

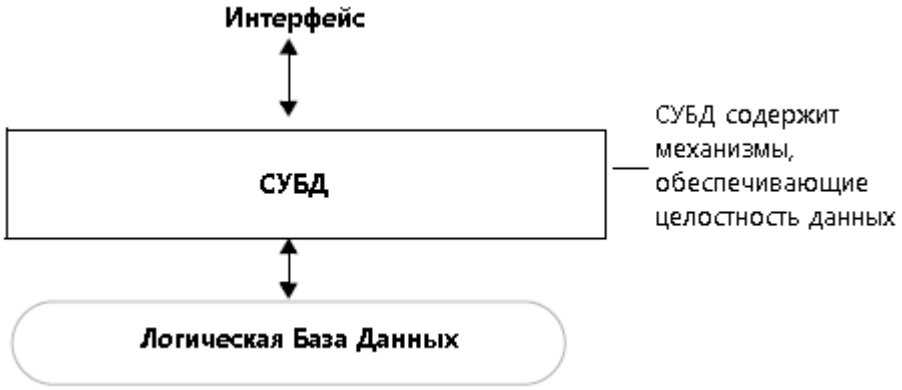
В данной главе будет рассказано, как Система Управления Базами Данных поддерживает целостность данных в многопользовательском окружении. В ней описано как среда C/SIDE обрабатывает ситуации, когда несколько пользователей или процессов пытаются изменить один объект.

Данная глава включает следующие разделы:

- Обеспечение целостности данных в многопользовательском окружении
- Блокировки в Dynamics NAV – сравнение двух вариантов серверов

27.1 Обеспечение Целостности Данных в Многопользовательском Окружении

Целостность данных связана с достоверностью данных, хранимых в базе данных, которые должны как можно точнее описывать реальный мир. Весь доступ к данным осуществляется с помощью СУБД (Системы Управления Базами Данных), ниже на рисунке изображен данный процесс:



СУБД управляет взаимодействием пользователя с базой данных, чтобы гарантировать, что ряд ограничений связанных с целостностью базы данных будут соблюдены. Соблюдая данные ограничения СУБД, защищает данные от повреждения и искажения. СУБД является очень сложной составляющей системы, но используемые ею принципы, направленные на поддержку целостности данных основываются на нескольких базовых концепциях:

- Транзакции записи и Восстановление (write transactions and recovery)
- Непротиворечивость чтения и Конкуренция (read consistency and concurrency)
- Блокировка таблиц (table locking)
- Выявление взаимных блокировок (deadlock detection)
- Фиксация обновлений (committing updates)

Эти концепции будут рассмотрены в следующих разделах.

Транзакции Записи и Восстановления (Write Transactions and Recovery)

Транзакции записи в C/SIDE являются минимальной единицей работы базы данных, которая может либо полностью выполнена, либо не выполнена совсем. Другими словами, транзакция это способ инкапсулировать последовательность операций чтения и записи в базе данных, чтобы гарантировать, что будет выполнены все операции, либо ни одна операция не будет выполнена. Концепция транзакций записи используется как однопользовательских инсталляциях, так и в многопользовательском окружении.

Когда транзакция отправляется СУДБ, система должна обеспечить, что:

- Все операции транзакции будут выполнены полностью и их результат зафиксирован в базе данных.

или

- Транзакция не окажет никакого влияния на базу данных.

СУБД должна предотвращать появление ситуаций, когда часть операций из транзакции будет применена к базе данных, а вторая часть не будет. Такая ситуация возможно, если при выполнении транзакции произошел сбой.

Некоторые типовые причины сбоя транзакции:

- Пользователь решил прекратить транзакцию.
- Транзакция не может быть завершена, т.к. не хватает какой-либо информации.
- Сбой системы из-за ошибок в программном или аппаратном обеспечении.
- Ошибка в операции, например, переполнение или деление на ноль.

Если транзакция была отменена, все таблицы возвращаются в состояние, в котором они находились на начало транзакции. Типичный пример транзакции записи это перевод 1000 рублей с одного счета на другой. Это приведет к появлению двух операций в одной базе данных:

1. Списать 1000 рублей со счета А.

2. Зачислить 1000 рублей на счет В.

Если произойдет нарушение в электроснабжении или другая неустранимая ошибка прервет выполнение программы после первой операции, база данных не будет находиться в согласованном состоянии, т.к. вторая операция не была завершена. Объединение операций в одну транзакцию гарантирует, что либо будут выполнены обе операции, либо ни одна операция не будет выполнена, в любом случае будет обеспечена целостность данных.

Дополнительно о Транзакциях Записи

В предыдущем разделе было сказано, что целостность данных будет обеспечена независимо от того была ли транзакция завершена или отменена. Способ, которым среда C/SIDE обрабатывает транзакции записи и поддерживает целостность базы данных, отличается от традиционных подходов. Традиционно, СУБД автоматически вносят в лог файл информацию об измененных записях. Этот лог файл содержит образы записи «перед» и «после» изменениями. Информация об изменениях, записанная в лог файл может использоваться для восстановления базы данных при возникновении ошибки.

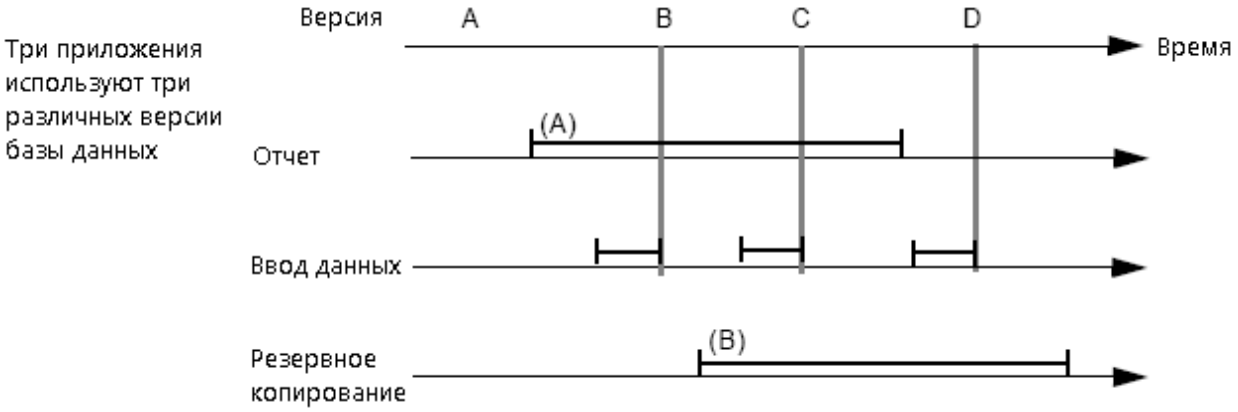
Предположим, что приложение прервалось из сбоя в электроснабжении или отменено оператором. База данных теперь находится в неопределенном состоянии, все сделанные изменения в базе данных должны быть отменены. В обычных СУБД это достигается с помощью, так называемого, *rollback recovery,* выполняется отмена всех сделанных изменений. Для возврата к предыдущему состоянию система читает лог файл назад и отменяет сделанные в базе данных изменения до тех пор, пока не достигнет точки начала транзакции. Таким образом, записям будут возвращены их исходные значения.

СУБД C/SIDE не требует лог файла потому, что база данных C/SIDE является *dataversion oriented*. Это означает, что каждый раз при фиксации транзакции создается новая версия базы данных. Пока вы вводите данные в базу данных, изменения являются частными до тех пор, пока изменения не будут зафиксированы. В этом случае изменения становятся общими, и выполняется создание новой версии. СУБД позволяет различным приложениям подключаться и изменять данных, для этого они получают индивидуальные версии базы данных, которые являются «снимками» (snapshot) базы данных на момент подключения приложения к базе данных. Преимущество подхода с использованием версий данных станет ясно после прочтения последующих разделов.

DRAFT

Непротиворечивость Чтения и Конкуренция (Read Consistency and Concurrency)

Среда C/SIDE использует механизм поддержания нескольких версий, это означает, что после выполнения транзакции записи создается новая версия базы данных. На следующем рисунке показано три приложения, работающие с одной базой данных. Представим, что первым подключился отчет, второе подключение – это пользователь, который вставляет новые записи в базу данных, третье подключение – процедура создания резервной копии.



В данном примере, создание отчета является времяемким процессом. Пока создается отчет, другой пользователь вводит или изменяет записи в базе данных. Как только запись фиксируется в базе данных, создается новая версия базы данных, но при запуске отчета был создан «снимок» (snapshot), и отчет продолжает работать с версией базы данных «A». Третий пользователь запустил процесс создания резервной копии. Когда было запущено создание резервной копии, был создан «снимок» (snapshot) текущей версии базы данных «B». Над этой версией и выполняется резервное копирование, не подвергаясь воздействиям со стороны второго пользователя, который продолжает вводить данные в систему. Работа с версиями данных позволяет одновременно нескольким пользователям подключаться к базе данных, не влияя друг на друга.

Наиболее важное преимущество от использования версий данных, является то, что разные приложения могут читать разные версии одной базы данных. Версия данных – «снимок» (snapshots) базы данных, сделанный при подключении к базе данных. Таким образом, СУБД обеспечивает непротиворечивость чтения и конкуренцию. Если в процессе доступа осуществляется только получение данных и не выполняются никакие изменения, то у всех приложений будет новая версия данных. Пока не будет выполнена транзакция записи, новая версия данных создаваться не будет. После обновления базы данных, частные изменения становятся общими. Зафиксированные обновления плюс часть базы данных, которая осталась неизменной дадут новую версию.

Что Такое Версии Данных (Data Version)?

Данные в базе данных хранятся в структуре данных, которая похожа на дерево. Структура данных дерево состоит из узлов. У каждого узла в дереве, кроме специального узла – корня, есть один родительский узел, и один и более дочерний узел. У корня нет родительских элементов. Узлы, у которых нет дочерних элементов, называются листьями, узлы, не являющиеся листьями, называются внутренними узлами. Уровень узла определяется как уровень его родительского узла плюс один, уровнем корня является ноль.

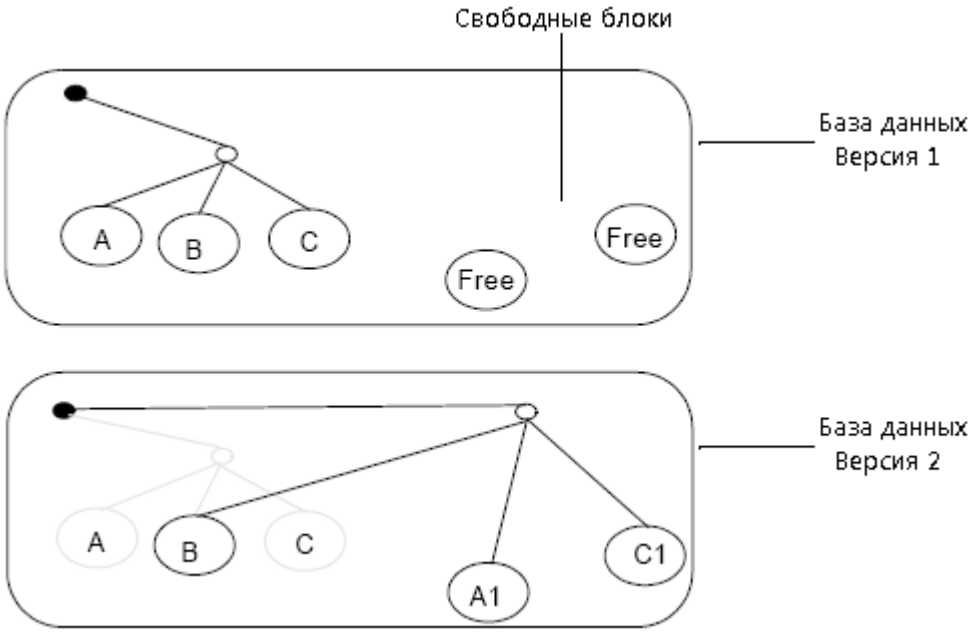
Структура данных, используемая в среде C/SIDE, называется Б+ дерево. Это означает, что структура дерева является сбалансированной и что данные (записи) хранятся только в листьях, но не во внутренних узлах. Преимуществом сбалансированного дерева является то, оно всегда содержит минимальное количество уровней, поэтому пути поиска будут настолько короткими, насколько это возможно. Структура Б+ дерева позволяет выполнять быстрый поиск.

Пример

DRAFT

Предположим, что древовидная структура в вашей базе данных, содержит ветвь с клиентами A, B и C, также там доступно два свободных блока.

Предположим, что нужно изменить клиентов A и C. При выполнении обновления записей, СУБД создаст копии оригиналов. Как показано на рисунке, копии используют два свободных блока базы данных. Затем выполняется изменение копий, также система создает новый внутренний узел.



Если при выполнении транзакции произойдет ошибка или она будет отменена пользователем, блоки занятые копиями будут освобождены и станут доступны для новых обновлений базы данных. Если транзакция зафиксирована, то новый внутренний узел заменит старый узел, а блоки, занятые старыми версиями клиентов A и C, станут доступны как свободные блоки и могут быть использованы для обновления баз данных.

В базе данных содержится ряд исторических версий. По мере того как свободные блоки заполняются успешными историческими версиями, новые версии начинают замещать старые.

У медленных операций с этим могут возникнуть проблемы. Предположим, что приложение A, при выполнении времяемкого отчета выполняет чтение данных. Тем временем, приложение B начинает выполнять транзакции записи, которые состоят из записей-заказов.

Новые версии базы данных будут создаваться по мере добавления записей-заказов в базу данных. Максимальное количество исторических версий в вашей базе данных зависит от неиспользуемого пространства в базе данных:

Максимальный размер базы данных

–

Использовано в текущий момент для хранения новых версий

=

Доступно для исторических версий

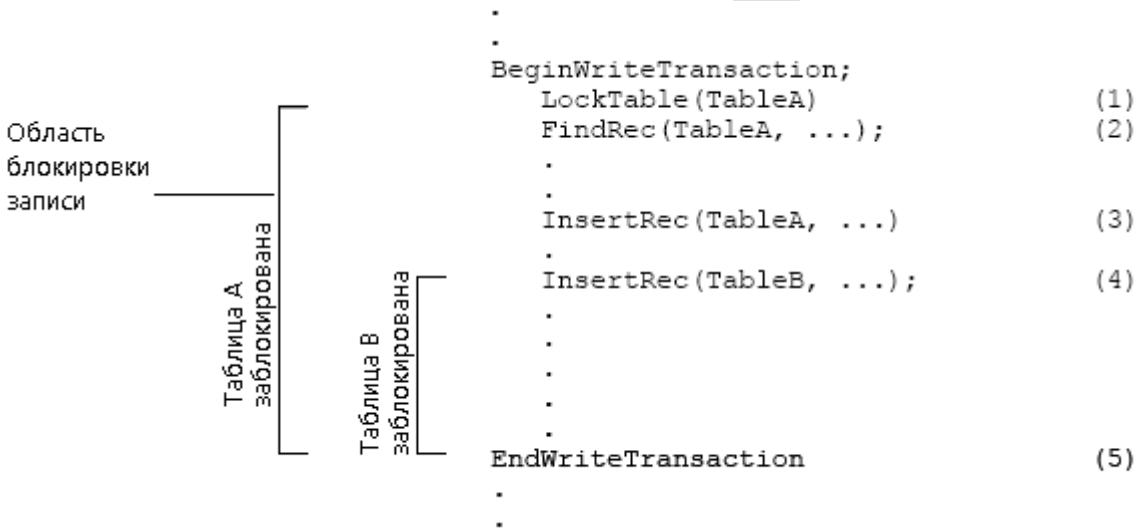
Через некоторое время, версия данных, используемая приложением A, станет самой старой версией. Приложению B для выполнения модификаций потребуются блоки, занимаемые данной версией. СУБД разрешит данный конфликт в пользу транзакций записи. У приложения A появится ошибка выполнения с текстом «Data version is no longer valid», теперь процесс нужно запустить повторно. Но пока приложение B продолжает работу, у приложения A мало шансов завершить формирование отчета. (Данную проблему можно решить увеличением размера базы данных)

Что Такое Блокировка Таблиц?

В многопользовательском окружении, чтобы обеспечить целостность данных, СУБД устанавливает блокировку записи на все таблицы, которые вы обновляете. Это позволяет предотвратить изменение тех же таблиц другими пользователями.

При выполнении транзакций записи, автоматически выполняется блокировка обновляемых таблиц, но также можно установить блокировку на таблицу даже если операция записи не выполняется. Блокируя таблицы непосредственно перед получением записи, вы гарантируете, что данные, которые вы собираетесь изменить, не будут изменены кем-либо еще, хотя с момента получения записи могло пройти какое-то время. Блокировка записи не влияет на получение данных. Это означает, что блокировка не мешает пользователям читать данные из той же таблицы.

В C/SIDE Database Server блокировка записи будет активна до тех, пока транзакция записи не будет завершена или отменена. На схеме ниже с помощью псевдо языка показана область блокировки записи.



Выше показаны автоматические и ручные блокировки. Строка (1) в транзакции записи блокирует таблицу A. Если бы ручная блокировка не была установлена на таблицу A, СУБД автоматически заблокировала бы данную таблицу при вставке записи (3). Таблица B вручную не блокируется, но блокируется автоматически при вставке записи (4). Обе блокировки остаются активными до выполнения команды EndWriteTransaction в строке (5).

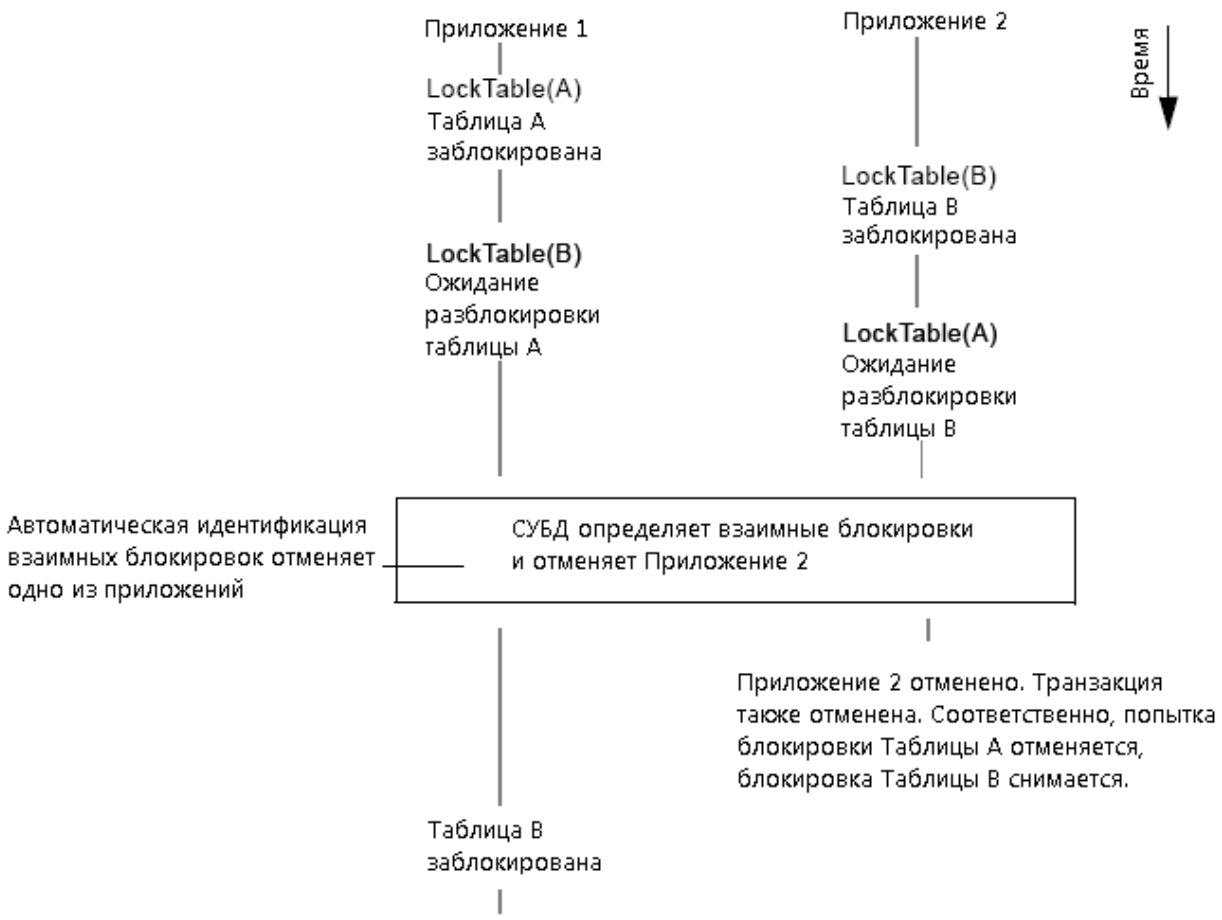
Что Такое Выявление Взаимных Блокировок?

Блокировка таблиц должна быть точно скоординирована, чтобы корректно работать в многопользовательском окружении.

Если различным транзакциям требуется доступ на запись в несколько таблиц сразу, то нужно быть очень аккуратным, чтобы избежать возникновения взаимных блокировок (*Deadlock*). Взаимная блокировка возникает, когда одна транзакция блокирует нужные ей таблицы, также вторая транзакция блокирует другие нужные ей таблицы, затем ни одна транзакция не может продолжать работу, т.к. она ждет, когда завершится вторая транзакция и освободит заблокированные ею таблицы.

Это означает, что каждая транзакция должна ждать, когда вторая транзакция завершится. В результате оба процесса будут ждать окончания друг друга бесконечно долго.

Чтобы избежать взаимных блокировок, в СУБД встроен механизм определения блокировок, который позволяет выявлять такие ситуации и отменять одну из транзакций записи. На следующем рисунке показано, как можно вызвать взаимную блокировку:



СУБД всегда отменяет приложение, которое вызвало взаимную блокировку. Это правило применяется всегда независимо от того сколько приложений было вовлечено во взаимную блокировку.

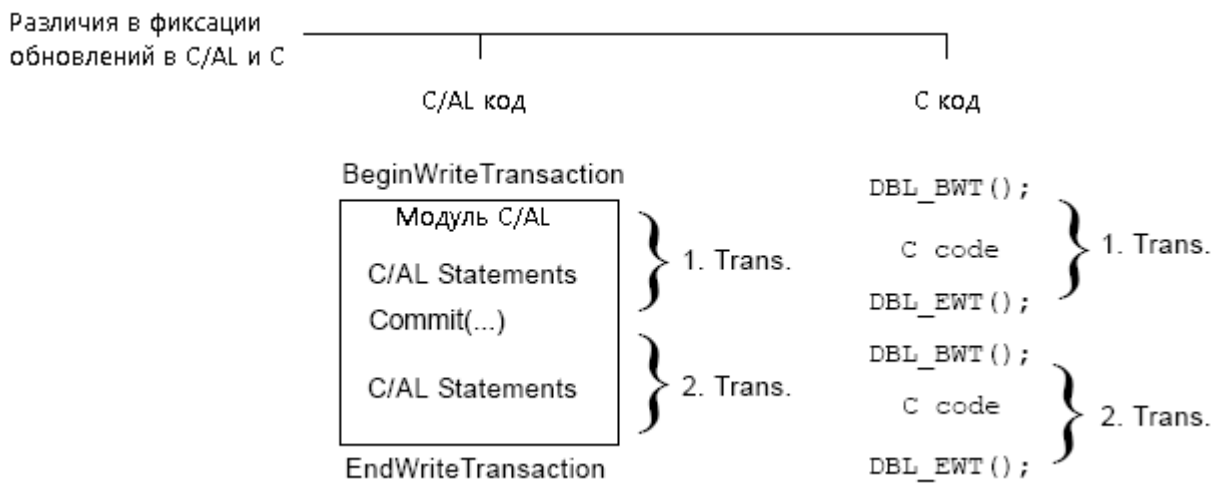
Есть ли Разница между Commit в языке C/AL и в C?

Хотя концепция фиксации обновлений одинакова как при использовании C/AL, так и при использовании C/FRONT (инструмент, который позволяет разрабатывать приложения, работающие с базой данных C/SIDE, с использованием языка программирования C), существует несколько незначительных отличий. В данном подразделе они будут описаны детально.

При использовании C/FRONT для выполнения обновлений нужно явно указать системе, что вы собираетесь выполнить транзакцию записи (используется DBL_BWT, BeginWriteTransaction.). Таким же образом, нужно использовать DBL_EWT (EndWriteTransaction) чтобы явно указать системе, что транзакция записи завершена.

При выполнении обновления базы данных C/SIDE с помощью C/AL операторы BeginWriteTransaction и EndWriteTransaction обрабатываются неявно. Система автоматически выполняет данные команды перед выполнением C/AL кода. Это значит, что если требуется выполнить одну транзакцию записи, фиксировать обновления явно не требуется – это будет сделано системой автоматически. Однако если нужно выполнить несколько транзакций записи, то для разделения транзакций нужно использовать функцию COMMIT().

На следующей схеме показаны отличия:



Код C/AL включает две транзакции записи. Когда начинается выполнение C/AL кода, автоматически начинается транзакция записи. С помощью функцию COMMIT()системе указывается, что первая транзакция записи завершена. Затем автоматически начинается вторая транзакция записи, которая завершается после выполнения C/AL кода. При использовании C, код выполняет те же транзакции, но каждая транзакция должны быть заключена между командами DBL_BWT() и DBL_EWT().

27.2 Блокировки в Dynamics NAV – Сравнение Двух Вариантов Серверов

В данном разделе будут описаны общие черты и различия в способах выполнения блокировок в двух вариантах серверов для Dynamics NAV: C/SIDE Database Server и SQL Server Option.

> **Важно**
>
> Данная информация относится только к типу транзакции UpdateNoLocks, используемому по умолчанию в SQL Server Option для Dynamics NAV. Дополнительную информацию об остальных типах транзакций см. в онлайновой справке *C/SIDE Reference Guide*.

Оба Варианта Серверов

Блокировки

В начале транзакции, считываемые данные не блокируются. Это означает, что чтение данных не вызывается конфликта с транзакциями, модифицирующими те же данные. Если нужно гарантировать, что из таблицы считываются самые последние данные, перед тем как читать данные нужно заблокировать таблицу.

Блокировка одной записи

Обычно, перед чтением записи ее блокировка не требуется, даже если затем она должна быть изменении или удалена. При попытке удалить или изменить запись, будет получено сообщение об ошибке, что в это время другая транзакция изменила или удалила запись. Данное сообщение будет получено потому, что среда C/SIDE проверит отметку времени (timestamp), которая хранится у всех записей базы данных, и выяснит, что отметка времени считанной копии отличается от отметки времени у модифицированной записи в базе данных.

Блокировка набора записей

Обычно, перед считыванием из таблицы набора записей, которые позже будут изменены или удалены, таблица блокируется. Блокировать таблицу нужно для того, чтобы гарантировать, что в это время другая транзакция не изменит записи. Если не заблокировать таблицу, то сообщение об ошибке не будет получено, даже если с момента чтения записи были изменены другой транзакцией.

DRAFT

Минимизация Взаимных Блокировок

Чтобы минимизировать количество взаимных блокировок, нужно выполнять блокировку записей и таблиц в одинаковом порядке для всех транзакций. Также можно попытаться выявить те области, где блокируется таблиц и записей больше, чем это необходимо, и попытаться уменьшить протяженность блокировок или полностью их убрать. Это позволит избежать возникновения конфликтов, связанных с данными записями и таблицами. Если это не исключило появления взаимных блокировок то, в качестве крайней меры, можно заблокировать больше таблиц и записей, чтобы отключить конкуренцию при выполнении транзакции

Если не удалось избежать появления взаимных блокировок с помощью программирования, нужно запускать проблемные транзакции по отдельности.

Блокировки в C/SIDE Database Server

При чтении из одного «снимка» версии (snapshot) данные не блокируются. При вызове функций выполняющих изменения (например, INSERT, MODIFY или DELETE) блокируется таблица целиком.

Блокировка набора записей

Как было написано выше, таблицы обычно блокируют, перед чтением из них данных, которые позже будут удалены или изменены. Для C/SIDE Database Server можно указать блокировку таблицы после того как были прочитаны данные (LOCKTABLE(TRUE,TRUE)), вместо того, чтобы блокировать таблицу перед чтением из нее данных (LOCKTABLE).

При попытке изменить или удалить записи, будет получено сообщение об ошибке, если другая транзакция в это время изменила транзакции. Также сообщение об ошибке будет показано, если в набор записей была вставлено новая запись. Однако если другая транзакция удалит запись из набора, то вы не сможете заметить это изменение. Целью блокировки с помощью LOCKTABLE(TRUE,TRUE) после того как были считаны данные, является улучшение конкуренции с помощью отложенной блокировки таблиц.

Блокировки в SQL Server

Если данные считываются без блокировки, то из базы данных будут получены последние данные (возможно незафиксированные). Если вызвать функцию Rec.LOCKTABLE, то сразу ничего не произойдет. Однако если после использования LOCKTABLE прочитать данные из таблицы, то данные заблокируются.

При вызове функций INSERT, MODIFY или DELETE, заданные записи блокируются немедленно. Это значит, что две транзакции, которые вставляют, изменяют или удаляют разные записи в одной таблице, не конфликтуют. Кроме того, блокировка будет установлена, если после использования для таблицы функций, изменяющих данных, будет выполнено чтение данных.

При вызове функций INSERT, MODIFY или DELETE, блокируется только одна запись, если для таблицы нет SumIndexFields или если при вызове данных функций не нужно обновлять никакие поля SumIndexFields.

Также важно заметить, что хотя изначально SQL Server устанавливает блокировку на одну запись, он также может решить использовать блокировку таблицы. Это происходит если, он определяет, что общая производительность возрастет, если не устанавливать блокировки на каждой записи. Улучшение производительности должно перевешивать потерю конкуренции, что происходит при использовании расширенных блокировок.

При чтении определенной записи, например, с помощью Rec.GET, данная запись блокируется. Это означает, что две транзакции, которые читают определенные, но разные записи в одной таблице, не будут конфликтовать.

При получении записей из набора записей, например, с помощью Rec.FIND('-') и Rec.NEXT, набор записей блокируется, по мере его считывания. Однако блокировки, внедренные в SQL Server, также

DRAFT

блокируют одну запись перед набором записей и одну после. Это означает, что две транзакции, считывающие разные наборы записей в таблице, вызовут конфликт, если между этими наборами записей нет записей. Когда блокировка устанавливается на набор записей, вторая транзакция не может установить блокировку ни на одну запись внутри набора.

Обратите внимание, что среда C/SIDE определяет, сколько записей должно быть получено от сервера при первой или следующей записи в наборе записей. Среда C/SIDE обрабатывает последующие чтения без дополнительных усилий, и отправляет на сервер меньшее количество запросов, что улучшает производительность. Кроме улучшения производительности это приводит к тому, что нельзя точно предсказать, когда будет установлена блокировка.

SQL Server Option для Dynamics NAV поддерживает для функции LOCKTABLE только значения по умолчанию– LOCKTABLE(TRUE,FALSE).

Различие Блокировок в Коде

Типичное использование LOCKTABLE(TRUE,TRUE) в C/SIDE Database Server показано в первом столбце таблицы. Эквивалентный код для SQL Server Option показан во втором столбце. Код, который работает на обоих серверах, показан снизу. Свойство *RECORDLEVELLOCKING* используется, чтобы определить, когда используется блокировка на уровне записей. В настоящее время только SQL Server Option для Dynamics NAV поддерживает блокировку на уровне записей.

C/SIDE Database Server

```
IF Rec.FIND('-') THEN
  REPEAT
  UNTIL Rec.NEXT = 0;
Rec.LOCKTABLE(TRUE,TRUE);
IF Rec.FIND('-') THEN
  REPEAT
    Rec.MODIFY;
  UNTIL Rec.NEXT = 0;
```

SQL Server

```
Rec.LOCKTABLE;
IF Rec.FIND('-') THEN
  REPEAT
  UNTIL Rec.NEXT = 0;
IF Rec.FIND('-') THEN
  REPEAT
    Rec.MODIFY;
  UNTIL Rec.NEXT = 0;
```

Оба варианта серверов

```
IF Rec.RECORDLEVELLOCKING THEN
  Rec.LOCKTABLE;
IF Rec.FIND('-') THEN
  REPEAT
  UNTIL Rec.NEXT = 0;
IF NOT Rec.RECORDLEVELLOCKING THEN
  Rec.LOCKTABLE(TRUE,TRUE);
IF Rec.FIND('-') THEN
  REPEAT
    Rec.MODIFY;
  UNTIL Rec.NEXT = 0;
```

DRAFT

ГЛАВА 28 ОБЗОР ФУНКЦИОНАЛЬНОСТИ CAPTION CLASS

Данная глава содержит описание функциональности caption class и объяснение как с данной функциональностью работает триггер CaptionClassTranslate (ID 15) в программном модуле 1.

Данная глава содержит следующие разделы:

- Обзор Функциональности Caption Class

28.1 Обзор Функциональности Caption Class -

Если для поля или элемента управления определено значение в свойстве *CaptionClass*, то каждый раз при отображении поля или элемента управления, вызывается триггер CaptionClassTranslate (ID 15) из программного модуля 1 (ApplicationManagement). Цель данной функции заменить существующий заголовок поля или элемента управления, на другую строку.

Синтаксис данной процедуры:

CaptionClassTranslate (<LANGUAGE>;<CAPTIONEXPR>)

LANGUAGE

<DataType> := [Integer]

<DataValue> := .....

CAPTIONEXPR

<DataType> := [String]

<Length> <= 80

<DataValue> := <CAPTIONAREA>,<CAPTIONREF>

Как можно видеть, в функцию передаются два параметра:

- LANGUAGE
- CAPTIONEXPR

LANGUAGE

Параметр LANGUAGE автоматически заполняется системой в виде Windows Language ID, на основании активного языка Dynamics NAV.

Пример

Если активным языком Dynamics NAV является English (United States), то в параметре LANGUAGE будет содержаться значение 1033. Если активным языком Dynamics NAV является Русский (Россия), то в параметре LANGUAGE будет содержаться значение 1049.

CAPTIONEXPR

В параметре CAPTIONEXPR содержится значение свойства *CaptionClass* поля или элемента управления.

Пример

В таблице 13 (***Salesperson/Purchaser***), поле **Global Dimension 1 Code (5050)** в свойстве CaptionClass содержит значение '1,1,1'. Откройте форму ***Salespeople/Purchasers***, активируйте отладчик и откройте окно **Все Поля** (Ctrl + F8) триггер CaptionClassTranslate будет показан в отладчике:

DRAFT

```
Codeunit 1 ApplicationManagement
Rounding := ReadRounding()
GLSetup.GET;
EXIT(GLSetup."Amount Rounding Precision");
CaptionClassTranslate(Language,CaptionExpr)
CommaPosition := STRPOS(CaptionExpr,',');
IF (CommaPosition > 0) THEN BEGIN
  CaptionArea := COPYSTR(CaptionExpr,1,CommaPosition - 1);
  CaptionRef := COPYSTR(CaptionExpr,CommaPosition + 1);
  CASE CaptionArea OF
    '1' : EXIT(DimCaptionClassTranslate(Language,CaptionRef));
    '2' : EXIT(VATCaptionClassTranslate(Language,CaptionRef));
  END;
END;
EXIT('');
```

В окне ***Variables*** на закладке ***Locals*** можно увидеть, что параметр CaptionExpr содержит значение '1,1,1'.

Context: Codeunit 1 ApplicationManagement\CaptionClassTranslate(L

Name	Value	Type
Text019	0	Integer
Language	1033	Integer
CaptionExpr	"1,1,1"	Text[80]
CaptionArea	""	Text[80]
CaptionRef	""	Text[80]
CommaPosition	0	Integer

All | Locals | Globals | Text Constants

Примечание

В окне ***Все Поля*** вы не найдете поля **Global Dimension 1 Code**. Вместо этого там будет поле **Department Code**– результат свойства *CaptionClass* и триггера CaptionClassTranslate.

Код Функции

Триггер CaptionClassTranslate (ID 15) является системным. Программист может внести в него свой код, который будет обрабатывать свойство ***CaptionClass***.

Взглянем на код в данном триггере в стандартной базе данных CRONUS:

```
CaptionClassTranslate(Language : Integer;CaptionExpr : Text[80])
CommaPosition := STRPOS(CaptionExpr,',');
IF (CommaPosition > 0) THEN BEGIN
  CaptionArea := COPYSTR(CaptionExpr,1,CommaPosition - 1);
  CaptionRef := COPYSTR(CaptionExpr,CommaPosition + 1);
  CASE CaptionArea OF
    '1' : EXIT(DimCaptionClassTranslate(Language, CaptionRef));
    '2' : EXIT(VATCaptionClassTranslate(Language, CaptionRef));
  END;
```

DRAFT

```
END;
EXIT('');
```

Данный код анализирует параметр CaptionExpr. Синтаксис данного параметра следующий:

CAPTIONEXPR := <CAPTIONAREA>,<CAPTIONREF>

В зависимости от значения CAPTIONAREA, будут вызываться различные процедуры. Посмотрите на оператор CASE … OF, будет вызываться либо функция:

```
DimCaptionClassTranslate(Language, CaptionRef)
```

либо

```
VATCaptionClassTranslate(Language, CaptionRef)
```

CAPTIONAREA

CAPTIONAREA это первая часть параметра CaptionExpr расположенная до первой запятой. Ее синтаксис:

```
CAPTIONAREA
  <DataType> := [SubString]
  <Length> <= 10
  <DataValue> := 1..9999999999
 // 1 для Измерений
 // 2 для НДС
```

> **Примечание**
>
> В стандартной функциональности определено только два значения CAPTIONAREA: 1 для Измерений и 2 для НДС.

CAPTIONREF

Вторая часть параметра CaptionExpr, которая идет после первой запятой – это CAPTIONREF, он передается в соответствующие функции целиком, а уже в них разбирается на элементы, на основании которых возвращаются заголовки полей и столбцов.

> **Примечание**
>
> Таким образом, стандартная функциональность NAV работает со свойством *CaptionClass*. Чтобы добавить новую функциональность, программист должен определить синтаксис и добавить код, в триггер CaptionClassTranslate (ID 15), чтобы обрабатывать данный синтаксис.

DRAFT

ГЛАВА 29 ПОДДЕРЖКА СИСТЕМЫ БЕЗОПАСНОСТИ НА УРОВНЕ ЗАПИСИ

В данной главе будет рассказано, как применять систему безопасности на уровне записи в SQL Server Option для Dynamics NAV.

- Система безопасности на уровне записи

29.1 Система Безопасности на Уровне Записи

SQL Server Option для Dynamics NAV позволяет ограничить доступ пользователей к информации, разрешив доступ только к тем записям таблицы, которые удовлетворяют определенным условиям. Это так называемая система безопасности на уровне записи, она активируется при указании фильтров защиты (security filters) в ролях и правах доступа, назначенных пользователю.

Важным моментом, который нужно иметь в виду при использовании фильтров защиты, является то, что фильтры и бизнес-логика приложения должны быть совместимы, иначе возможно возникновение проблем. Система безопасности на уровне записи не влияет на бизнес-логику приложения, а бизнес-логика может оказывать значительное влияние на систему безопасности на уровне записей и устанавливаемые фильтры. Безопасность на уровне записей полезна только в том случае, когда она признается/поддерживается бизнес-логикой.

Внедрение Системы Безопасности на Уровне Записи

После того как были настроены фильтры защиты, Dynamics NAV будет применять их автоматически в большинстве ситуаций. Это значит, что пользователь получит сообщение об ошибке только в том случае, если попытается вручную получить данные за пределами диапазона, ограниченного фильтрами защиты.

При вызове формы с помощью командной кнопки или элемента меню, среда C/SIDE автоматически применит фильтры защиты к главной переменной типа запись(record), используемой на форме (предполагается, что для запуска формы использовались свойства командной кнопки или элемента меню, а не код).

Аналогично, при вызове отчета или датапорта с помощью командной кнопки или элемента меню, среда C/SIDE автоматически применит фильтры защиты к переменным типа запись (record) используемым на форме запроса (закладки-фильтры).

Среда C/SIDE не применяет фильтры защиты к глобальным или локальным переменным, определенным пользователем. Чтобы помочь пользователю оставаться внутри диапазона, определенного фильтрами защиты в код нужно включить соответствующую функцию, которая применит фильтры в коде. Для этого используется функция SETPERMISSIONFILTER, которая доступна для переменных типа запись (record).

Пример

В данном примере будет выполнен поиск первой записи в диапазоне, определенном фильтрами защиты:

```
Customer.SETPERMISSIONFILTER
Customer.FIND('-')
```

Примечание

Фильтры безопасности влияют на производительность точно также как любые другие фильтры, установленные пользователем. Важно, чтобы в таблицах с большим количеством записей, для фильтров защиты существовали подходящие ключи, и чтобы эти ключи использовались. Среда C/SIDE не выполняет автоматический выбор наиболее подходящего ключа. Например, если установлен фильтр защиты, позволяющий пользователю видеть

DRAFT

только записи, созданные им самим (фильтр по полю **User ID)**, то в таблице должен существовать подходящий ключ, который начинается с User ID. Кроме того, чтобы обеспечить лучшую производительность, данный ключ должен быть установлен при открытии формы. В некоторых случаях для этого нужно изменить сортировку используемую формой по умолчанию, или изменить свойства в командной кнопке или элементе меню, или в коде, который открывает форму.

Формы, Отчеты и Датапорты

Как было сказано ранее, при запуске форм, отчетом и датапортов с помощью командных кнопок или элементов меню, среда C/SIDE автоматически применяет фильтры защиты (предполагается, что для запуска используются свойства командной кнопки или элемента меню, а не код). Это означает, что для простой формы применение фильтров защиты на уровне пользователя будет достаточно, чтобы обеспечить требуемый уровень безопасности.

Find(‘+’) и calcsums

Однако если в отчете или форме содержится более сложный код на языке C/AL, например, Find('+') или calcsums, то возможно, что установки фильтров защиты на уровне пользователя будет недостаточно, для успешного запуска формы или отчета. В данном случае, пользователь может получить сообщение об ошибке, которое указывает, что у пользователя нет прав для чтения определенной таблицы.

Данная ситуация обычно возникает потому, что форма или отчет содержит функциональность, для которой требуется доступ код всем записям таблицы, что вступает в конфликт с фильтрами защиты, которые предоставляют доступ пользователю только к некоторым записям таблицы. Например, такое может произойти, если перед вставкой новой записи, требуется получить последнюю запись таблицы, а она находится вне диапазона, определенного фильтрами защиты для пользователя.

Для преодоления данной сложности существует достаточно простой способ. Пользователю назначена роль, которая дает ему права на доступ к записям таблицы. У этой роли есть ограничения на записи, какой-либо таблицы, заданные фильтрами защиты, и эти ограничения конфликтуют с функциональностью формы. Чтобы разрешить конфликт нужно расширить права пользователя, назначив ему роль, которая дает косвенный (*indirect)* доступ на чтение всех записей таблицы.

Теперь, когда пользователь запустит форму, код сможет прочитать последнюю запись таблицы, даже если она находится за пределами диапазона, определенного фильтрами защиты для пользователя. Тот факт, что пользователю предоставлены косвенные права на всю таблицу не уменьшает целостность системы безопасности на уровне записей. Пользователь по прежнему не может просматривать записи за пределами диапазона, определенного фильтрами защиты, но приложение может выполнять расчеты и считывать все необходимые для корректной работы данные, и предоставлять пользователю всю требуемую информацию.

Данное решение работает потому, что права доступа складываются, а не взаимоисключаются.

Программный Модуль

Типичное приложение Dynamics NAV также включает формы и отчеты, содержащие сложную функциональность, требующую запуска одного или нескольких программных модулей. В данных программных модулях может содержаться код, который работает с несколькими различными таблицами.

В Dynamics NAV программные модули являются независимыми объектами, которые обладают собственными правами доступа. Им не назначаются роли, но указываются права на доступ к различным таблицам. Права доступа программного модуля является частью его свойств.

Когда пользователь запускает форму, вызывающую программный модуль, нет гарантий, что фильтры защиты определенные для пользователя будут совместимы с правами доступа заданными

DRAFT

для программного модуля. Если они несовместимы, пользователь получит сообщение об ошибке, которое указывает, что у пользователя нет прав для чтения определенной таблицы.

Чтобы разрешить данный конфликт нужно обеспечить, чтобы фильтры защиты, определенные для пользователя, были совместимы с правами доступа заданными для программного модуля. Для этого нужно расширить права пользователя, назначив ему роль, которая дает косвенный *indirect* доступ на чтение всех записей таблицы, затем открыть программный модуль в Object Designer, открыть окно ***Properties*** и задать для программного модуля права чтения на требуемую таблицу.

Однако сообщение может не содержать всю необходимую информацию. В данном случае следует использовать отладчик, чтобы выявить участок кода, где происходит ошибка. Это поможет идентифицировать задействованную таблицу и дать программному модулю права на чтение данной таблицы. Затем нужно снова запустить форму. Если вновь будет получено сообщение об ошибке, нужно повторять указанную процедуру до тех пор, пока форма не станет работать как следует.

Возможно, ваша программная лицензия не позволяет изменять требуемые программные модули. В этом случае нужно обратиться к партнеру Microsoft Business Solutions.

Руководство по Внедрению Системы Безопасности на Уровне Записи

Принимая во внимание все изложенное выше, можно сформулировать несколько общих правил, которыми следует руководствоваться при внедрении системы безопасности на уровне записи:

- Система безопасности на уровне записи и бизнес-логика приложения должны работать вместе.
- Для форм, отчетов и датапортов, которые запускаются с помощью свойств, а не кода, достаточно применения фильтров защиты, которые задаются на уровне пользователя.
- Для форм, отчетов и датапортов, которые выполняют код, требуется назначение пользователю дополнительных косвенных прав доступа на уровне пользователя.
- Для программных модулей, требуется назначение дополнительных косвенных прав доступа, как на уровне пользователя, так и на уровне объекта.
- Права доступа складываются, а не взаимоисключаются.

  Важно:

  Система безопасности на уровне записи не поддерживает символы «*» и «?» в фильтрах. Можно использовать любые символы: <, >, |, &,.. и =.

  Максимальная длина фильтра защиты составляет 250 символов, в расчет также принимаются символы <, >, |, &,.. и = (эти символы уменьшают длину фильтра защиты).

  Более того, фильтры защиты конкатенируются, и сумма всех фильтров защиты назначенных пользователю или роли не может превышать 250 символов.

Применение Фильтров Защиты при Учете

Как было упомянуто ранее, когда функциональность задействует один или более программных модулей, применение фильтров защиты требует определенной ловкости. Типичным примером является процедура учета. Фильтры защиты, определенные для пользователя обычным способом, должны сопровождаться дополнительными косвенными правами доступа, как на уровне пользователя, так и на уровне объекта.

В следующем примере будет показана сложность процесса установки фильтров защиты, ограничивающих доступ пользователя к определенной функциональности.

В данном примере подразумевается, что пользователь может учитывать G/L Journal (Фин. Журнал). Пользователю назначены следующие роли:

Роль	Описание

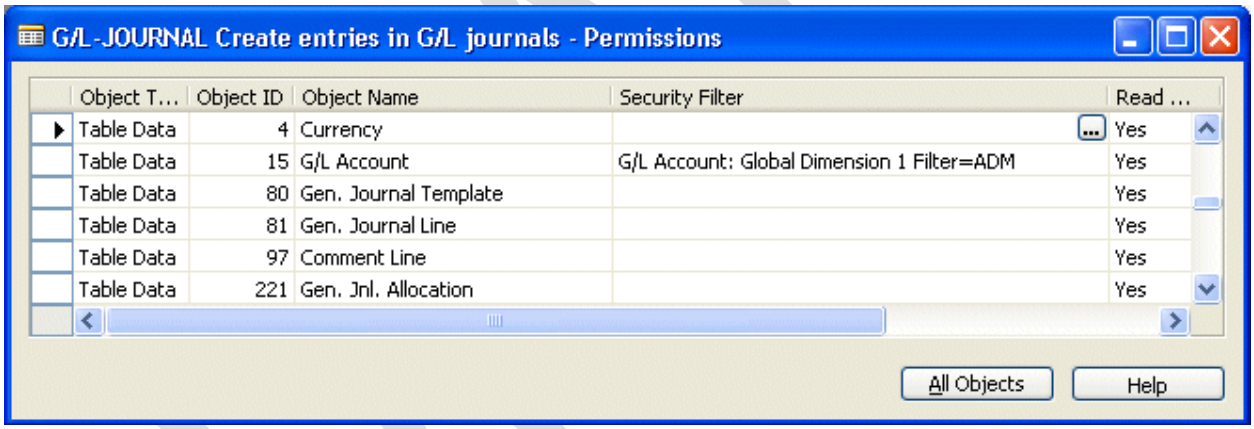
Роль	Описание
ВСЕ	Все пользователи
ФИН-ЖУРНАЛ	Созд. записей в фин. журналах
ФИН-ЖУРНАЛ, УЧЕТ	Учет фин. Журналов
Payment Terms	Роль создана для данного примера и дает права доступа к таблице ***Payment Terms***.

Первые три роли являются стандартными ролями Dynamics NAV, а последняя роль – это новая роль, созданная для данного примера. Для данного примера также был создан пользователь *TestUser*, которому были назначены данные роли.

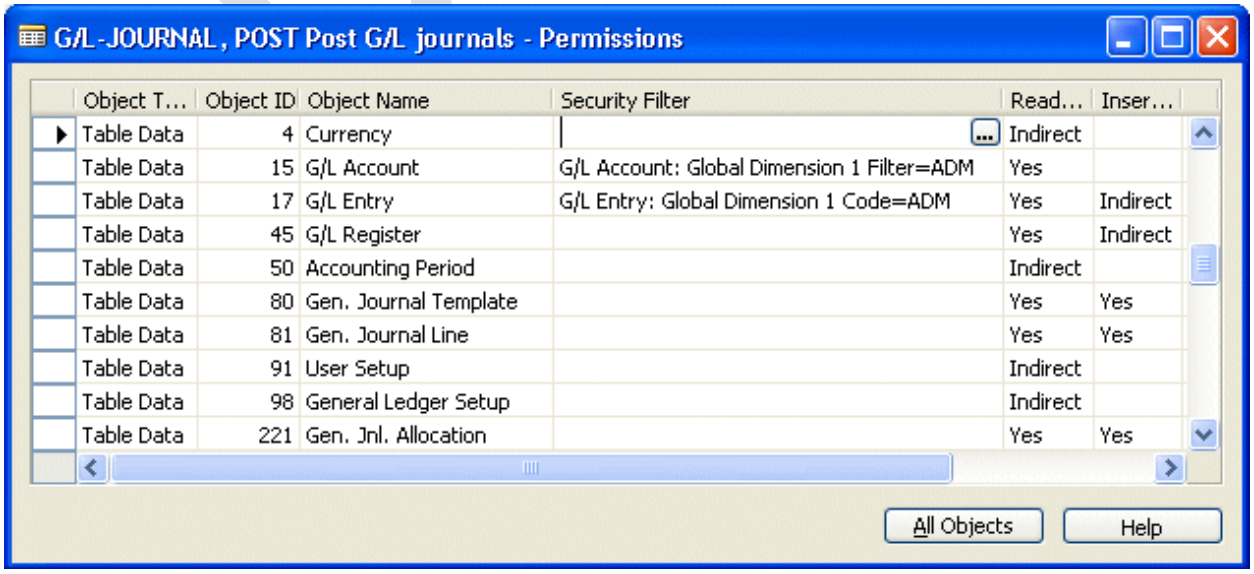
Примечание

Для выполнения данного примера нужно иметь возможность заходиться в систему под двумя разными пользователями: *TestUser*, который будет учитывать G/L Journal (Фин. Журнал), и Администратор, который будет изменять роли и права доступа, назначенные пользователю *TestUser*. Администратору должна быть назначена роль SUPER. Также важно помнить, что пользователь *TestUser* должен отключаться и снова входить в систему каждый раз, когда администратор изменяет систему прав доступа.

В данном сценарии требуется, чтобы пользователь *TestUser* мог учитывать записи, связанные с отделом Администрация (Global Dimension 1). Для этого нужно установить фильтры защиты для ролей, назначенных пользователю *TestUser*. Нужно применить данные фильтры ко всем ролям, которые предоставляют пользователю *TestUser* права на доступ к таблицам ***G/L Account*** и ***G/L Entry***:



Object T...	Object ID	Object Name	Security Filter	Read ...
Table Data	4	Currency		Yes
Table Data	15	G/L Account	G/L Account: Global Dimension 1 Filter=ADM	Yes
Table Data	80	Gen. Journal Template		Yes
Table Data	81	Gen. Journal Line		Yes
Table Data	97	Comment Line		Yes
Table Data	221	Gen. Jnl. Allocation		Yes



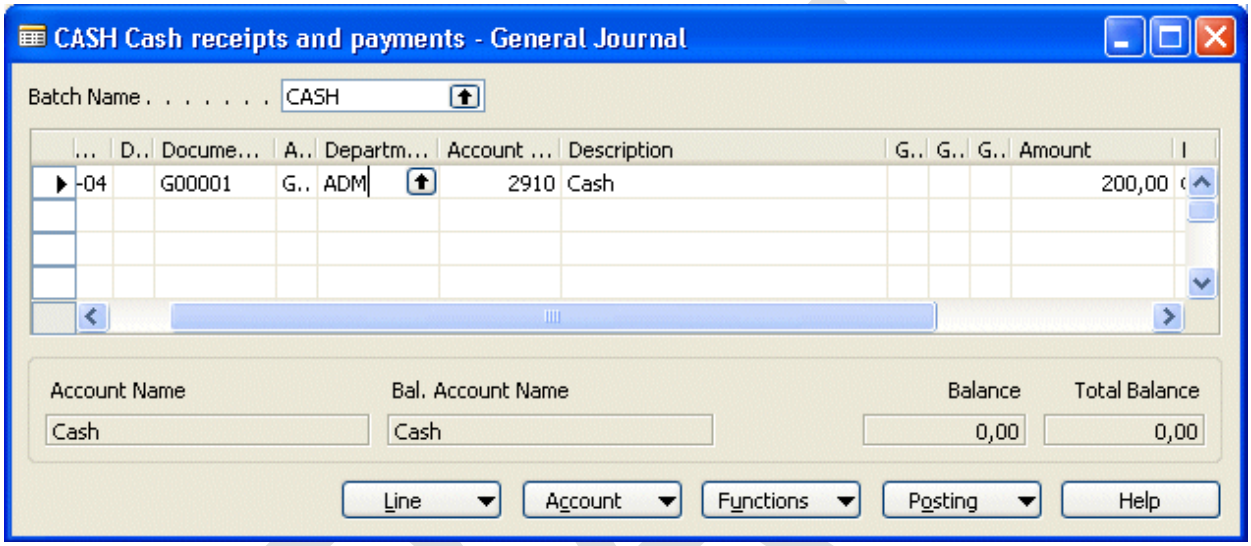
Object T...	Object ID	Object Name	Security Filter	Read...	Inser...
Table Data	4	Currency		Indirect	
Table Data	15	G/L Account	G/L Account: Global Dimension 1 Filter=ADM	Yes	
Table Data	17	G/L Entry	G/L Entry: Global Dimension 1 Code=ADM	Yes	Indirect
Table Data	45	G/L Register		Yes	Indirect
Table Data	50	Accounting Period		Indirect	
Table Data	80	Gen. Journal Template		Yes	Yes
Table Data	81	Gen. Journal Line		Yes	Yes
Table Data	91	User Setup		Indirect	
Table Data	98	General Ledger Setup		Indirect	
Table Data	221	Gen. Jnl. Allocation		Yes	Yes

Посмотрим, что произойдет, когда пользователь *TestUser* попытается учесть General Journal (Финансовый Журнал).

> **Примечание**
>
> Цель данного примера – убедится, что пользователь *TestUser* может учитывать только те записи General Journal, которые связаны с отделом Администрация. Поэтому в окне **Финансовый Журнал** (***General Journal)*** следует сделать поле **Отдел Код (Department Code)** видимым.

1. Войдите в базу данных как *TestUser* и откройте меню Финансовые Операции и щелкните Финансовые Журналы, чтобы открыть окно ***Финансовый Журнал (General Journal)***.

2. Введите информацию, которую вы хотите учесть. Помните, что в поле Отдел Код должно быть установлено значение АДМ:

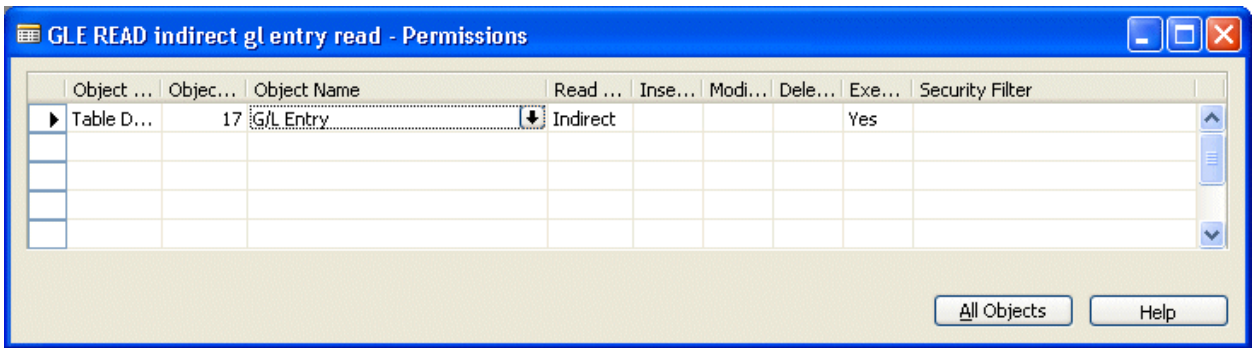


Если поле Отдел Код не видно, щелкните Вид, Показать Столбец, чтобы открыть окно Показать Столбец и установите флаг напротив поля Отдел Код.

3. Щелкните Учет, Учет (F11), чтобы учесть запись. Вы получите сообщение об ошибке, информирующее, что у вас нет прав для чтения таблицы ***G/L Entry (Фин. Книга Операций)***.

Это произошло из-за того, что каждый раз при учете новой записи ей должен присваиваться новый уникальный идентификатор. Перед тем как создать этот идентификатор, Dynamics NAV должен узнать идентификатор последней записи в таблице ***G/L Entry***. Если последняя запись таблицы не связана с отделом АДМ, то пользователю *TestUser* не будет разрешено считать данную запись, в результате появится сообщение об ошибке. Если последняя запись будет связана с отделом АДМ, пользователь *TestUser* сможет прочитать ее и не получит сообщения об ошибке.

Чтобы решить данную проблему нужно предоставить пользователю *TestUser* косвенные права на чтение всей таблицы ***G/L Entry***. Для этого, нужно войти в базу данных как администратор и создать новую роль, и назначить ее пользователю *TestUser*:

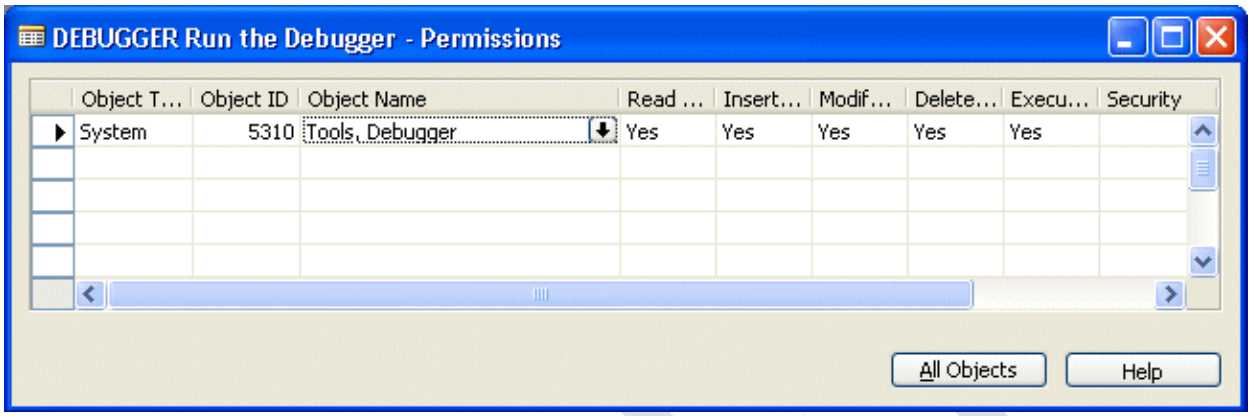


Не забывайте синхронизировать систему безопасности после изменения системы прав доступа.

Войдите в базу данных как *TestUser* и попытайтесь снова учесть запись.

К сожалению, снова будет получено тоже сообщение об ошибке, информирующее, что у вас нет прав для чтения таблицы ***G/L Entry (Фин. Книга Операций)***. Это произошло из-за того, что форма вызывает несколько программных модулей и одному или нескольким из этих программных модулей требуется считывать записи из таблицы ***G/L Entry***. Чтобы выяснить какой программный модуль вызывает ошибку нужно использовать отладчик.

Нужно дать пользователь *TestUser* права на запуск отладчика. Для этого, нужно войти в базу данных как администратор и создать новую роль, и назначить ее пользователю *TestUser*:



Не забывайте синхронизировать систему безопасности после изменения системы прав доступа.

Отладка кода

Чтобы выявить программный модуль, который вызывает ошибку нужно выполнить следующие шаги:

1. Войти в базу данных как *TestUser* и открыть окно Финансовый Журнал.

2. В финансовом журнале должна быть введена операция в поле Отдел Код, которой должно быть установлено значение АДМ. Пока не учитывайте данную операцию.

3. Щелкните Сервис, Debugger, Точка Останова на Триггерах, чтобы снять флаг с данного параметра, т.к. нужно, чтобы отладчик останавливался только при наступлении ошибки.

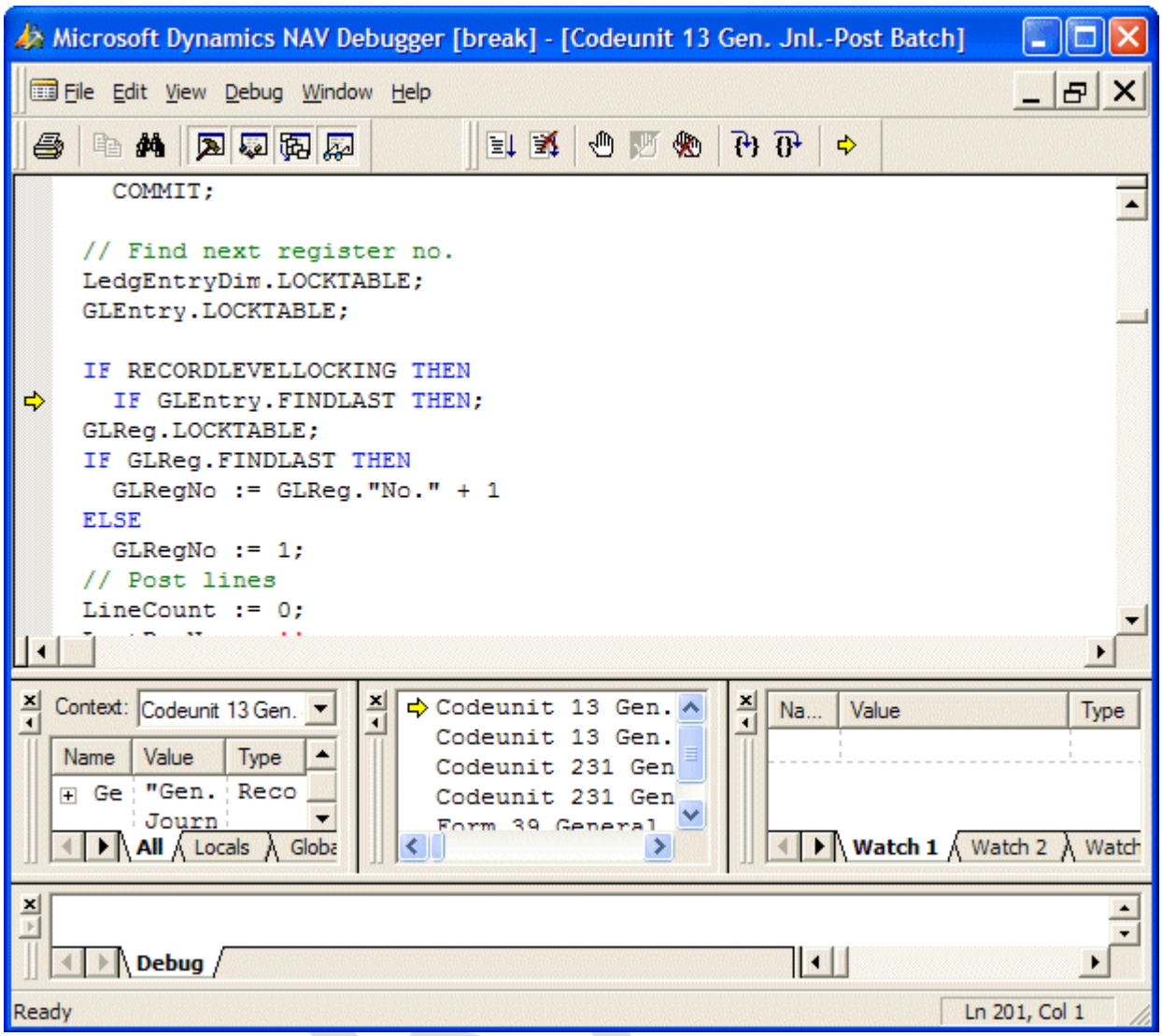
4. Щелкнуть Сервис, Debugger, Active, чтобы активизировать отладчик.

5. В окне Финансовый Журнал щелкнуть кнопку Учет, Учет (F11).

Приложение будет работать чуть медленнее, т.к. в фоновом режиме работает отладчик.

Через некоторое время отладчик откроется и покажет место в коде, где встретится первая ошибка:

DRAFT



Как видно на рисунке, первая ошибка возникла в программном модуле 13 ***Gen. Jnl.-Post Batch,*** где в коде была совершена попытка получить последнюю запись (функция FINDLAST) в таблице ***G/L Entry***.

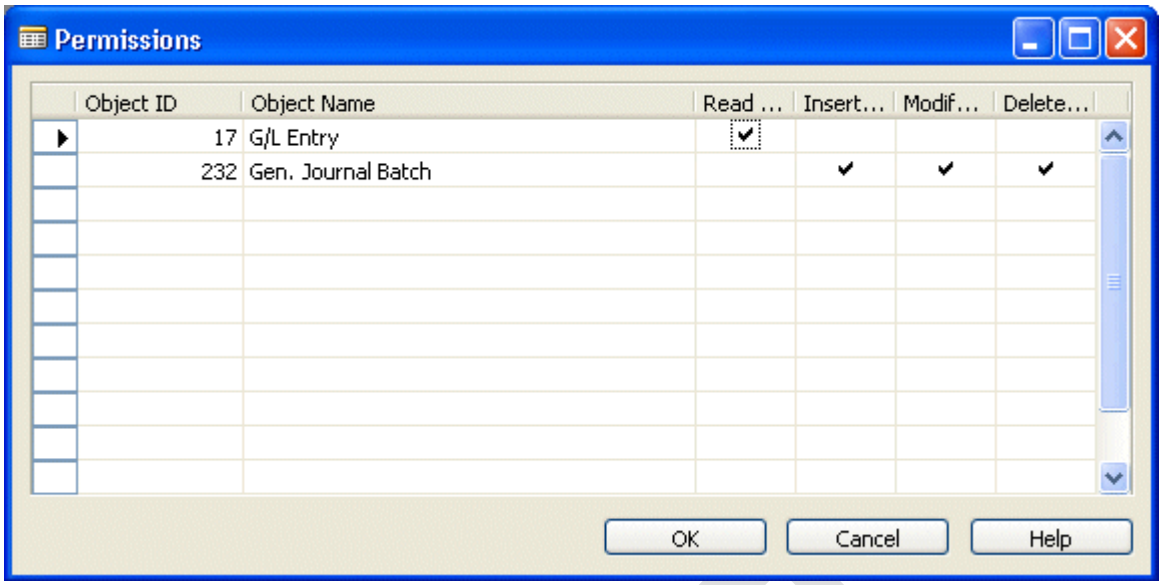
Чтобы устранить данную проблему нужно дать программному модулю 13 права на чтение всех записей в таблице ***G/L Entry***.

Чтобы дать программному модулю дополнительные права:

1. Открыть Object Designer, затем открыть программный модуль 13, ***Gen. Jnl.-Post Batch*** и открыть окно ***Properties***.

2. Щелкнуть кнопку AssistButton в свойстве *Permissions*.

3. Добавьте таблицу ***G/L Entry*** в список и убедитесь, что к данной таблице у программного модуля доступ только на чтение:



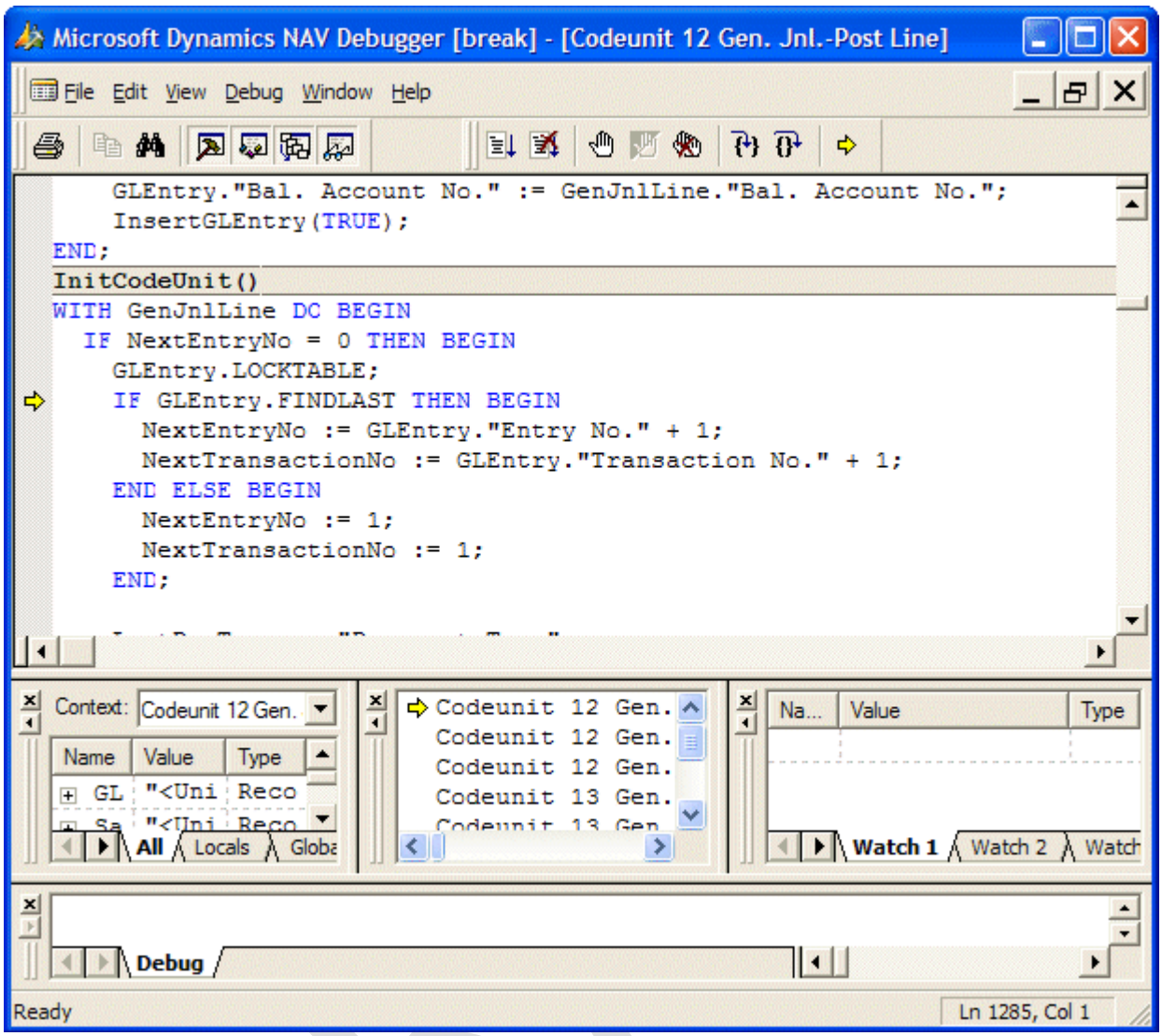
4. Щелкните кнопку OK, чтобы зафиксировать сделанные изменения.

5. Закройте, сохраните и откомпилируйте программный модуль.

После изменения свойств выполнять синхронизацию системы безопасности не требуется, т.к. программный модуль не является частью системы прав доступа.

Теперь войдите в базу данных как *TestUser*, активизируйте отладчик и попытайтесь снова выполнить учет. Произойдет еще одна ошибка:

DRAFT



В окне ***Debugger*** видно, что на этот раз ошибка произошла в программном модуле 12 ***Gen. Jnl.-Post Line,*** где в коде была совершена попытка получить последнюю запись (функция FINDLAST) в таблице ***G/L Entry***.

Чтобы устранить данную проблему нужно повторить процедуру, описанную выше, и дать программному модулю 12 ***Gen. Jnl.-Post Line*** права на чтение всех записей в таблице ***G/L Entry***.

Теперь войдите в базу данных как *TestUser*, активизируйте отладчик и попытайтесь снова выполнить учет. На этот раз учет будет выполнен корректно.

Выключите отладчик и попытайтесь учесть запись, связанную с другим отделом, и посмотрите, что произойдет. В этом случае будет отображаться другое сообщения, информирующее, что у вас нет прав на запись в таблицу ***G/L Entry***.

Вы знаете, что это не так потому, что вы только что учли операцию. Значит, ваши фильтры защиты работают, и вы можете учитывать только операции связанные с отделом АДМ.

Фильтры Защиты и Сложные Формы

Как было упомянуто ранее, среда C/SIDE не применяет фильтры защиты к локальным и глобальным переменным, определенным пользователем. Это означает, что если ваше приложение содержит сложные формы, которые используют переменные для доступа к таблицам, содержащим важные данные, то установленные фильтры защиты не будут применены к этим переменным. Чтобы гарантировать, что пользователи не смогут получить доступ к важным данным, нужно изменить код, содержащийся в формах.

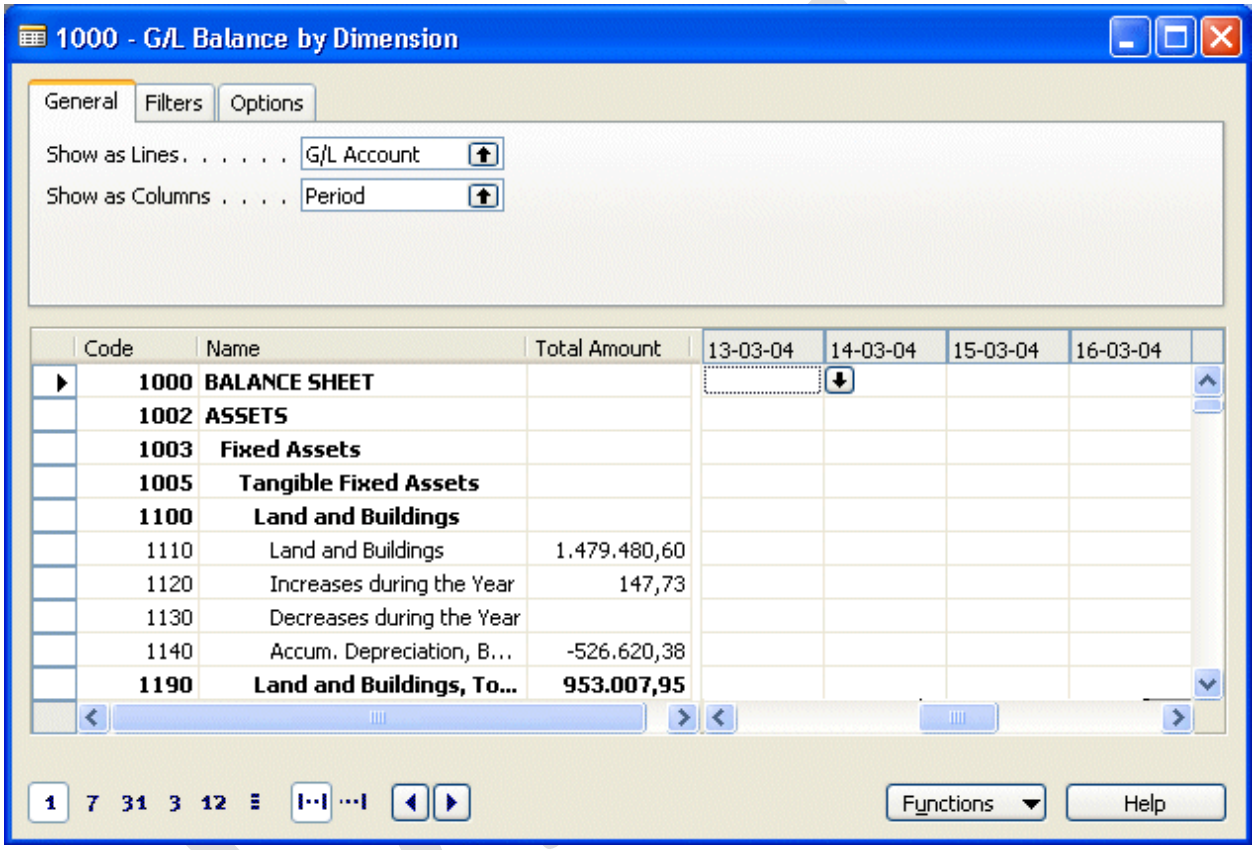
В предыдущем примере, пользователю *TestUser* было предоставлено право на учет операций только для отдела АДМ. Теперь нужно выяснить, может или нет, пользователь *TestUser* читать данные по другим отделам.

Войдите в базу данных как *TestUser* и откройте формы на базе таблицы General Ledger, чтобы проверить какие данные будут отображены. Это должны быть сложные формы, которые используют переменные.

Чтобы проверить одну из таких форм:

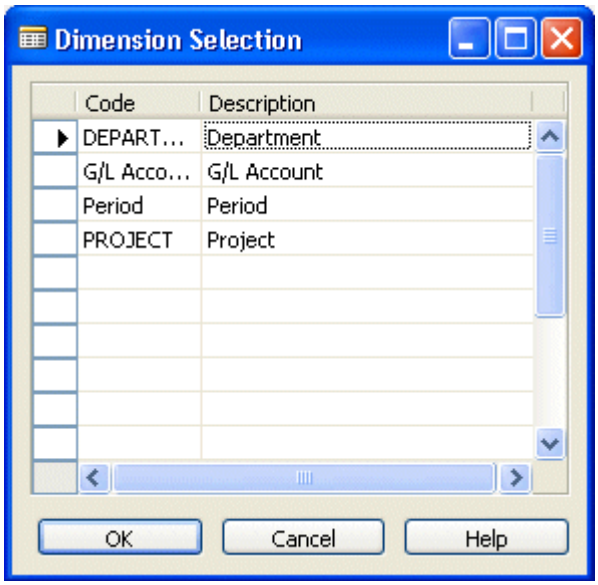
1. В окне **Финансовые Операции** щелкните План Счетов, чтобы открыть окно **План Счетов**. В данном окне будут отображены данные по отделу АДМ.

2. В окне План Счетов щелкните кнопку Баланс, Фин. Баланс по Измерению, чтобы открыть окно **Фин. Баланс по Измерению**:



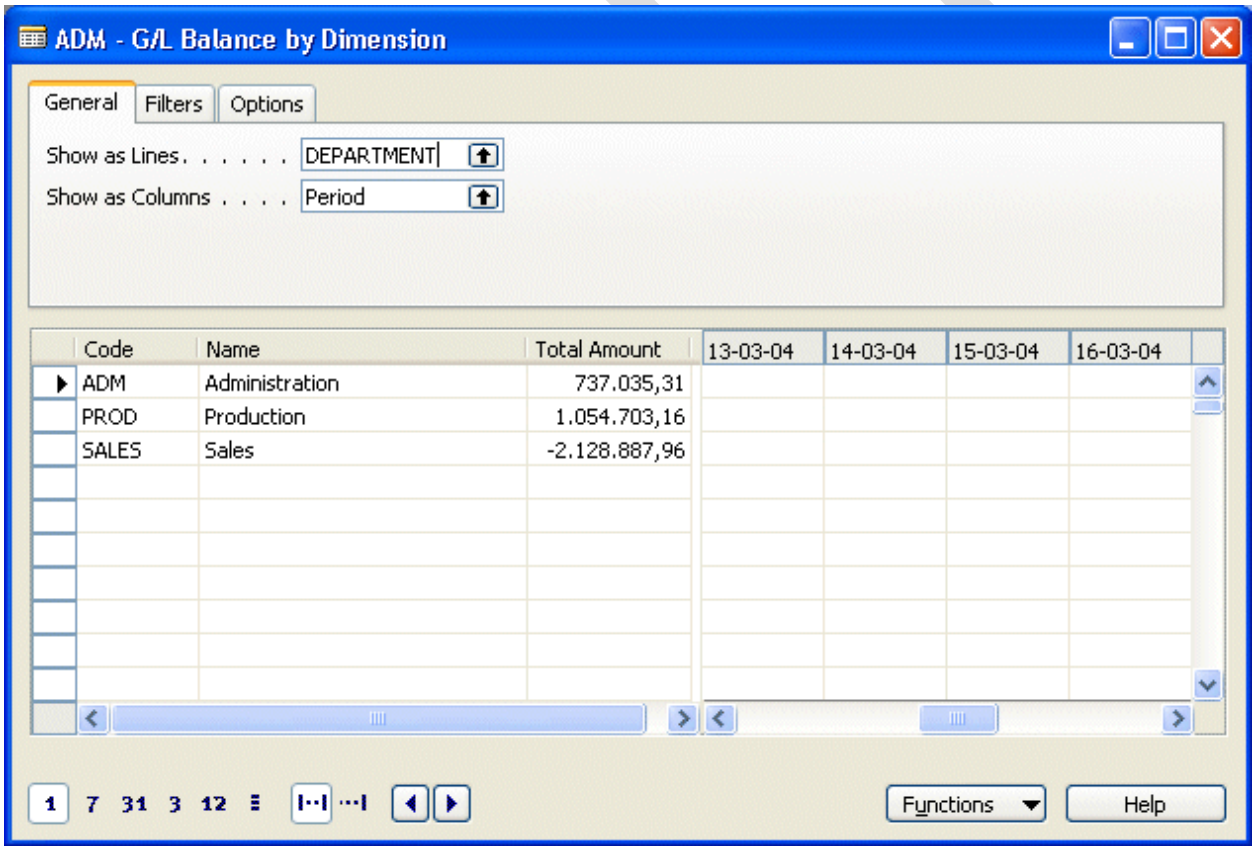
Это матричная форма, отображает итоговые суммы по всем счетам в плане счетов. Кроме того, того список балансов может быть представлен несколькими способами в зависимости от примененных фильтров.

3. Щелкните кнопку AssistButton в поле **Показывать в Строках**, чтобы открыть окно Измерение Выбор:



В данном окне можно выбрать различные измерения, по которым будут сгруппированы отображаемые в окне ***Фин. Баланс по Измерению***.

4. Выберите Отдел и щелкните кнопку OK, чтобы вернуться в окно ***Фин. Баланс по Измерению***:



Как можно видеть, в данном окне отображаются три итоговых суммы, по одной для каждого отдела.

Предполагалось, что пользователь*TestUser* сможет видеть итоги только по отделу Администрация. Но ситуация не настолько плоха как может показаться.

Щелкните кнопку Детализировать в поле **Общая Сумма** для отдела Продажи, чтобы открыть окно ***Фин. Книга Операций***. Окно будет пустым, это означает, что были применены фильтры защиты, и пользователь *TestUser* не может видеть финансовые операции по другим отделам.

Однако пользователю *TestUser* не полагается видеть итоги в окне Фин. Баланс по Измерению. Чтобы это исправить нужно, изучить код данной формы и определить, как его можно улучшить, чтобы он стал поддерживать фильтры защиты.

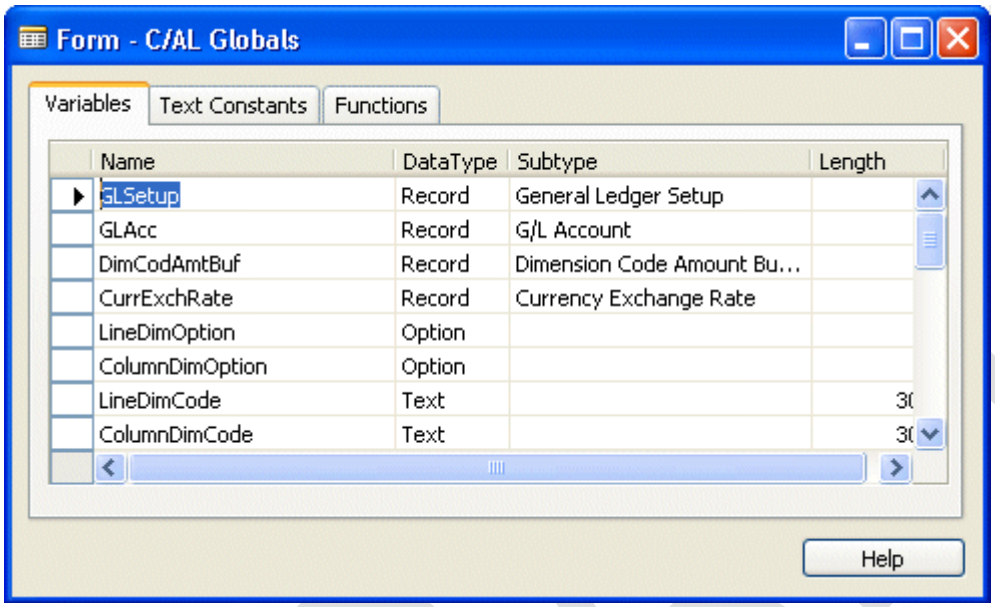
Просмотр кода

Чтобы исследовать содержащийся на форме код нужно:

1. Войти в базу данных как администратор и открыть окно **Фин. Баланс по Измерению**.

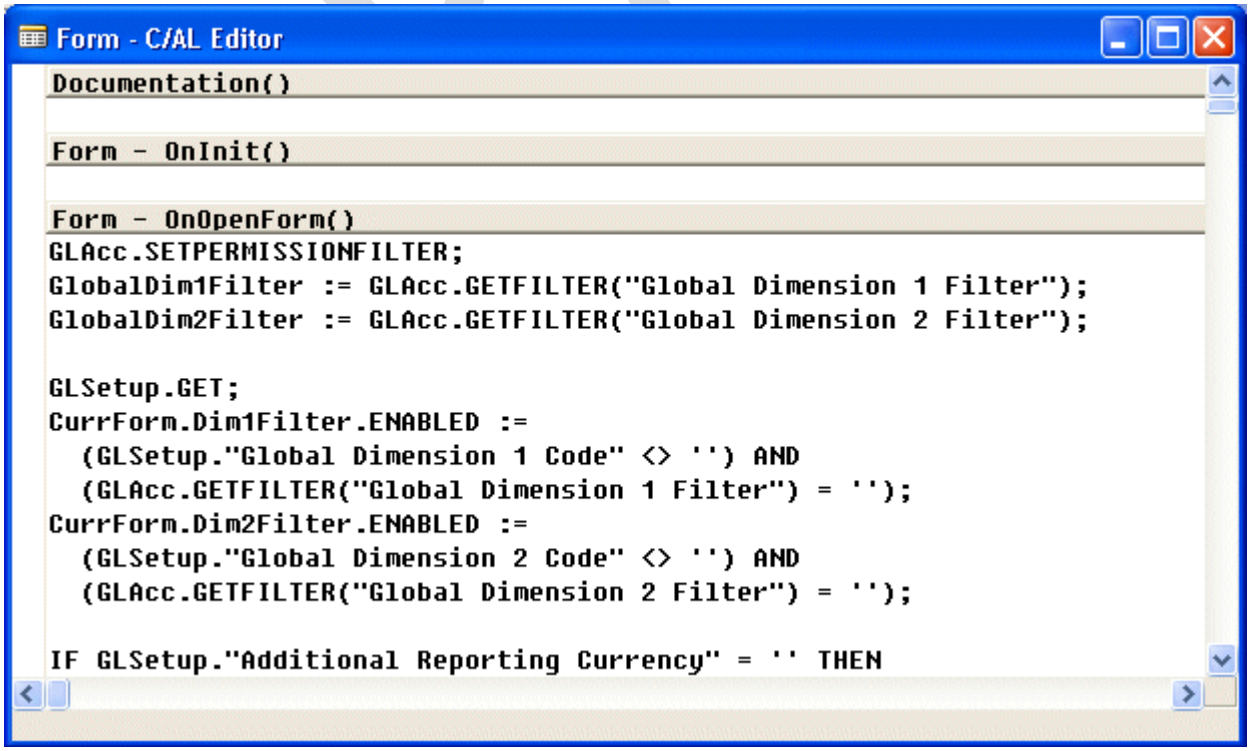
2. Нажать клавиши CTRL + F2, чтобы открыть форму в дизайнере.

3. Щелкнуть Вид, C/AL Globals, чтобы открыть окно ***C/AL Globals***:



В окне можно увидеть глобальную переменную *GLAcc*, которая ссылается на таблицу ***G/L Account***.

4. Щелкните Вид, C/AL Code, чтобы открыть C/AL Editor и посмотрите содержащийся на форме код:



Первая строка кода в триггере *OnOpenForm* содержит:

GLAcc.SETPERMISSIONFILTER

Это гарантирует, что все фильтры защиты, установленные для пользователя, будут применены к данной глобальной переменной.

Однако исследуемая матричная форма включает функциональность, позволяющую изменять установленные фильтры. Это подразумевает, что фильтры устанавливаются и сбрасываются. В языке C/AL есть функция RESET,которая используется для сброса всех фильтров.

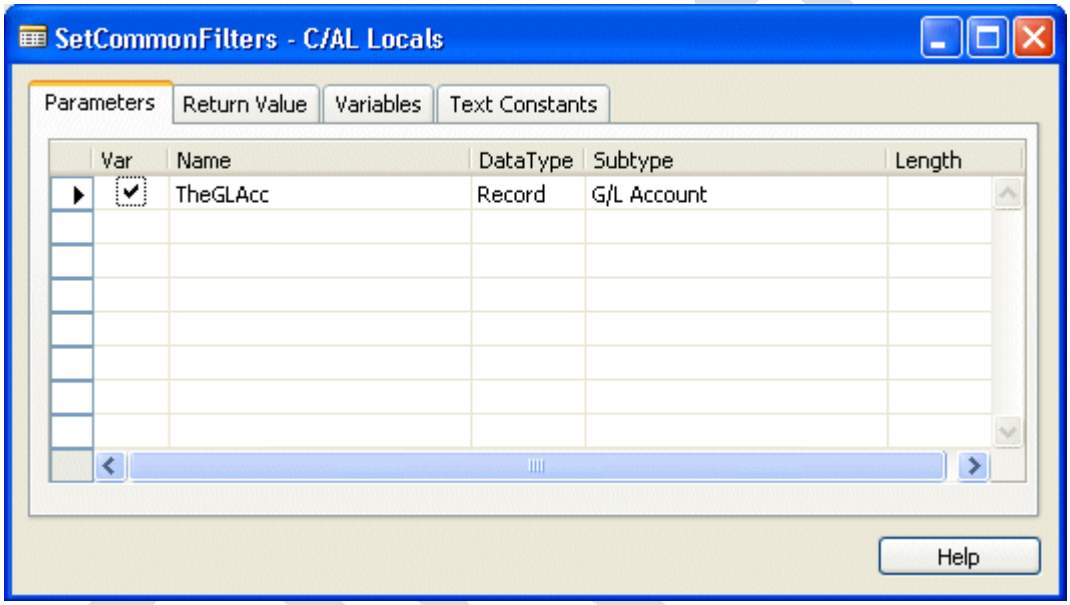
Принимая это во внимание, хорошей идеей будет посмотреть, не используется ли в коде формы ***G/L Balance by Dimension (Фин. Баланс по Измерению)*** функция RESET.

5. Используйте окно ***Найти***, чтобы найти все места, где используется функция RESET.

Она используется в двух местах: функции *SetCommonFilters* и функции *CalcAmount.* Обе функции являются пользовательскими.

6. Посмотрим локальные переменные, объявленные для данных функций.

Функция *SetCommonFilter* выглядит многообещающе:



Как можно видеть в нее по ссылке передается параметр, который связан с таблице ***G/L Account***.

7. Рассмотрим код данной функции внимательнее:

Form - C/AL Editor

```
SetCommonFilters(VAR TheGLAcc : Record "G/L Account")
CLEAR(TheGLAcc);
WITH TheGLAcc DO BEGIN
  IF DateFilter = '' THEN
    DateFilter2 := ExcludeClosingDateFilter
  ELSE BEGIN
    IF AmountType = AmountType::"Net Change" THEN BEGIN
      DateFilter2 := DateFilter;
    END ELSE BEGIN
      SETFILTER("Date Filter",DateFilter);
      DateFilter2 := STRSUBSTNO('..%1',GETRANGEMAX("Date Filter"));
    END;
    IF ExcludeClosingDateFilter <> '' THEN
      DateFilter2 := DateFilter2 + '&' + ExcludeClosingDateFilter;
  END;
  RESET;
  IF GLAccFilter <> '' THEN
    SETFILTER("No.",GLAccFilter)
  ELSE
    SETRANGE("No.");
  IF GLAccFilter <> '' THEN
```

TextBox - C/AL Editor

```
  PeriodInitialized := FALSE;

CalcAmount(SetColFilter : Boolean) : Decimal
IF SetColFilter THEN
  ColumnCode := CurrForm.Matrix.MatrixRec.Code
ELSE
  ColumnCode := '';
IF DimCodAmtBuf.GET(Code,ColumnCode) THEN
  EXIT(DimCodAmtBuf.Amount);

GLAcc.RESET;
SetCommonFilters(GLAcc);
SetDimFilters(GLAcc,0);
IF SetColFilter THEN
  SetDimFilters(GLAcc,1);
```

В функции *CalcAmount* сразу после кода, вызывающего функцию RESET для глобальной переменной *GLAcc*, выполняется вызов функции *SetCommonFilters*.

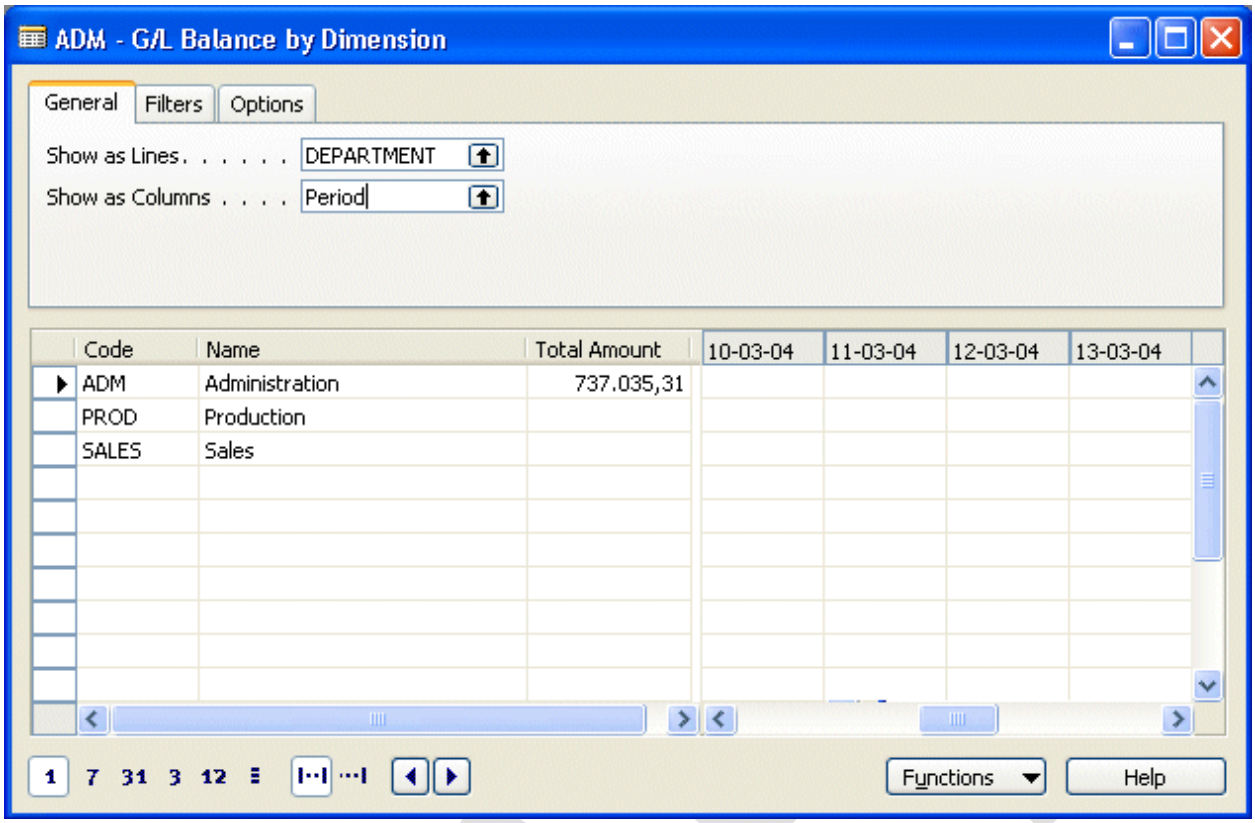
Все это намекает на то, что нам нужна функция RESET, вызываемая из функции *SetCommonFilters*.

8. Введите следующий код сразу после вызова функции RESET в функции *SetCommonFilters*:

GLAcc.SETPERMISSIONFILTER

9. Закройте, сохраните и откомпилируйте форму.

10. Войдите в базу данных как *TestUser* и откройте окно ***Фин. Баланс по Измерению***. Не забудьте выбрать значение Отдел в поле Показывать в Строках:



Данные, принадлежащие другим отдела, больше на форме не показываются.

Вы успешно внедрили систему безопасности на уровне записи в учетную процедуру и обеспечили доступ пользователя только к разрешенной информации, заданной фильтрами защиты.

DRAFT

ГЛАВА 30 ПРОИЗВОДИТЕЛЬНОСТЬ

В данную главе описаны функции среды C/SIDE предназначенные для увеличения производительности, сюда входят Кэш СУБД, Кэш Дисков и Командный Буфер. Также в ней содержится раздел о влиянии ключей и запросов на производительность.

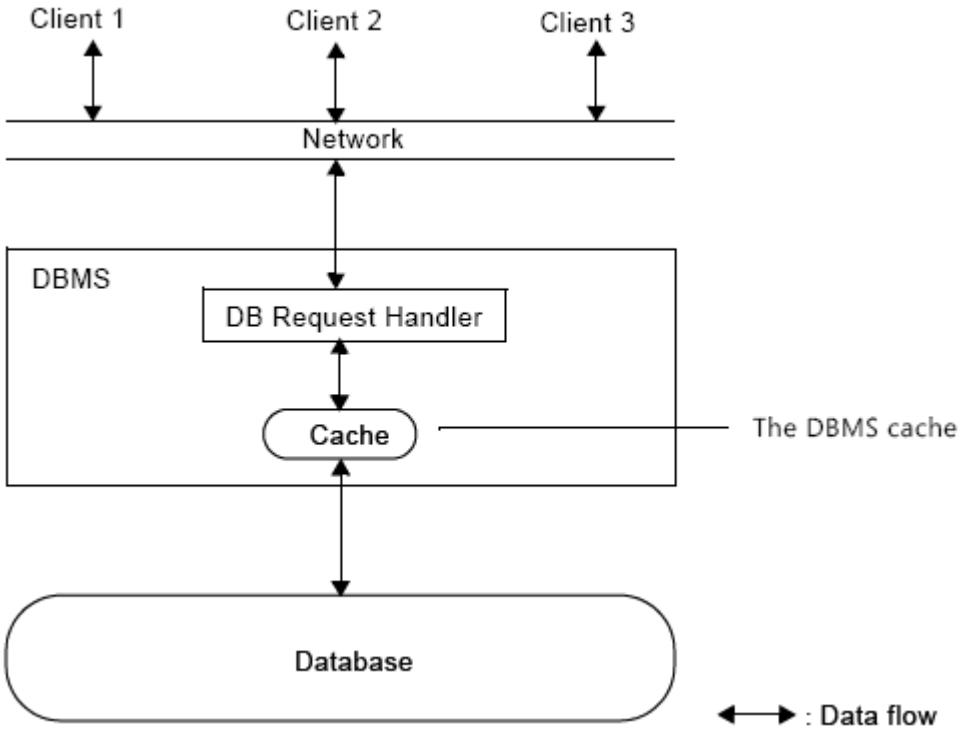
- Кэш СУБД (DBMS Cache)
- Кэш Дисков (Commit Cache)
- Командный Буфер (Command Buffer)
- Ключи, запросы и производительность
- Функции C/AL и производительность на SQL Server
- Хранимая Процедура Входа для SQL Server Option

30.1 Кэш СУБД (DBMS Cache)

Кэш СУБД это буфер памяти, в котором хранятся копии частей базы данных, используемых в текущий момент Системой Управления Базами Данных. Чтение из памяти выполняется на много быстрее чтения с диска. Таким образом, СУБД возвращается запись из памяти быстрее, чем, если бы ее потребовалось считать с диска. Если запрашиваемые данные хранятся в памяти, то они доступны сразу. Если требуемые данные в кэше не хранятся, то они вначале копируются с диска в кэш.

Кэш СУБД полностью скрыт от пользователя. Если пользователь запрашивает данные, они автоматически копируются в кэш. Если данные изменяются, они автоматически копируются на физический диск. Эти операции с данными выполняются автоматически. Пользователю не нужно ничего знать о кэше.

На следующем рисунке показаны три клиента, отправляющие запросы в СУБД. Когда, например, клиент 2 отправил запрос на чтение данных из базы данных, обработчик запросов определяет можно ли получить требуемые данные из кэша или нужно получить их с диска.



В это же время другой клиент может изменить запись в таблице базы данных. Измененная запись будет запись записана в кэш СУБД, а не на диск. Когда клиент завершит транзакцию записи

(зафиксирует изменения), данные из кеша, измененные в процессе транзакции, будут записаны на диск. Кэш постоянно обновляется соответствующей информацией из базы данных.

Размер кэша сильно влияет на производительность. При определении размера кэша нужно руководствоваться двумя простыми правилами:

- Чем больше памяти выделяется для кэша, тем эффективнее он становится (нет смысла делать кэш больше, чем размер вашей базы данных)
- Размер кэша не должен превышать физический объем памяти доступный в системе. В этом случае операционная система начнет сохранять кэш на диск, и считывать его обратно. Это значительно снизит общую скорость системы C/SIDE.

  Примечание

  Чтобы включить кэширование транзакций записи нужно использовать параметр сервера commitcache=yes. Подробнее в следующем разделе.

30.2 Кэш Дисков (Commit Cache)

Кэш дисков это специальный буфер для записи на диск. Кэш дисков разработан для:

- Быстрого получения зафиксированных транзакций от СУБД. Это освобождает СУБД для выполнения других задач.
- Включает асинхронную запись на диск.
- Включает параллельные операции чтения и записи, когда используется несколько дисков.
- Гарантирует непротиворечивость файла.

Кэш дисков помещается между СУБД и базой данных. Он получает зафиксированные транзакции от СУБД. После получения зафиксированных транзакций он начинает записывать данные на диск(и). Таким образом, СУБД может выполнять другие задачи, пока кэш дисков выполняет запись на диск. Поэтому запись на диск не выполняется одновременно с фиксацией транзакции.

Как было описано в разделе 4.3 Физическая и Логическая Базы Данных, логическая база данных может храниться в нескольких файлах (каждый из которых может храниться на отдельном диске). При использовании нескольких дисков для хранения базы данных, каждый диск управляется отдельным процессом Кэш Дисков, которые объединены вместе и позволяют осуществлять параллельные операции чтения и записи.

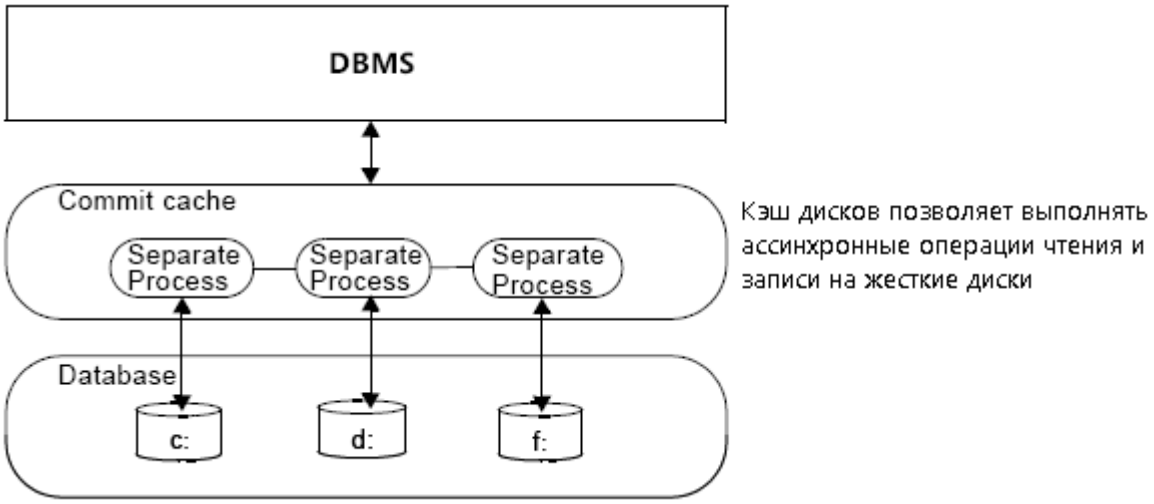
Кеш дисков обеспечивает непротиворечивость файла базы данных, даже если при выполнении операции записи произошел сбой в электроснабжении (правда в этом случае будут потеряны все зафиксированные транзакции, которые в тот момент хранились в кэше дисков).

Примечание

Не стоит использовать программы кэширования дисков, с отложенной записью. Использование таких систем кэширования может повредить файл базы данных.

На следующей схеме изображена база данных, которая хранится на трех физических дисках. Каждый диск управляется своим процессом Кэш Дисков. Объединенные вместе, эти процессы позволяют выполнять параллельно операции чтения и записи.

DRAFT



30.3 Командный Буфер

Командный буфер используется только для C/SIDE Database Server, и находится между вашим приложением и СУБД. Это временное хранилище, которое содержит запросы (команды на языке C/AL), отправленные от вашего приложения в СУБД. Командный буфер был создан, чтобы уменьшить количество передач по сети, при использовании среды C/SIDE в локальной сети.

При выполнении транзакций записи, некоторые запросы, такие как вставка записи в таблицу (record.INSERT()), не обязательно должны сразу отправляться в СУБД. Они могут временно храниться в командном буфере. В целом, если команды не должны возвращать значения, то они могут не отправляться на сервер сразу.

> **Примечание**
>
> Содержимое командного буфера отправляется в СУБД либо когда он будет заполнен, либо если команда требует от СУБД немедленного ответа.

Преимуществом компоновки команд СУБД в пакеты, является уменьшение количества операций по сети, что уменьшает загрузку локальной сети. Это происходит потому, что время необходимое для отправки одного запроса в СУБД сопоставимо со временем отправки целого пакета.

Следующий код демонстрирует, как командный буфер влияет на количество операций по сети.

```
WHILE Record.FIND('-') DO
  Record.DELETE();
```

Для каждой записи таблицы выполняется две команды. Однако для каждой записи выполняется только один запрос к СУБД. Дело в том, что команда DELETE храниться в командном буфере пока не будет выполнена команда FIND.

Отладка

Система автоматически отключает командный буфер при активации отладчика. Это может приводить к некоторой путанице, если не знать об этом факте.

Следующий код (содержимое воображаемого программного модуля), иллюстрирует различия при выполнении кода с отладчиком и без:

```
Customer."No." := '12';
Customer.DELETE();
First := 7;
Second := 0;
```

DRAFT

```
Ratio := First / Second;
```

Если клиента с номером 12 не существует, то ошибка возникнет в любом случае, независимо от того активен отладчик или нет. Однако ошибки будут различаться. В данном коде две ошибки: если клиент не найден, то функция DELETE не сможет быть выполнена. Кроме того в последней строке выполняется деление на ноль.

Если отладчик выключен, то команда DELETE будет храниться в командном буфере, чтобы выполниться позже. Поэтому возникнет ошибка выполнения из-за деления на ноль.

Если отладчик активен, то команда DELETE будет выполнена немедленно. В результате произойдет ошибка выполнения из-за того, что клиент с номером 12 не найден.

30.4 Ключи, Запросы и Производительность

При написании запроса, который выполняет поиск по набору записей в таблице, чтобы Dynamics NAV мог выполнить запрос быстро, важно правильно определить ключи, причем как в таблице, так и в запросе. Например, записи операций для одного клиента, являются маленьким набором записей таблицы, содержащей записи по всем клиентам.

Если Dynamics NAV сможет найти и прочитать набор эффективно, что выполнение запроса будет зависеть только от размера набора. Если Dynamics NAV не сможет эффективно найти и прочитать набор, то производительность ухудшится. В наихудшем случае Dynamics NAV будет читать все записи в таблице, а не только в подходящем наборе. В таблице с 100,000 записей ответ на запрос может занять от пары миллисекунд до нескольких секунд.

Чтобы улучшить производительность нужно в таблице определить ключи, которые будут способствовать выполняемым запросам. Эти ключи должны быть корректно указаны в запросах.

Например, нужно получить записи, связанные с определенным клиентом. Чтобы это сделать нужно применить фильтр на поле **Customer No.** в таблице ***Cust. Ledger Entry***. Чтобы эффективно выполнить запрос на SQL Server в таблице должен быть определен ключ, начинающийся с поля **Customer No.**. Также нужно указать данный ключ в запросе.

Ключи таблицы:

1. Entry No.
2. Customer No.,Posting Date

Запрос может выглядеть следующим образом:

```
SETCURRENTKEY("Customer No.");
SETRANGE("Customer No.",'1000');
IF FIND('-') THEN
  REPEAT
  UNTIL NEXT = 0;
```

При использовании C/SIDE Database Server также нужно указывать ключи в запросах. Однако C/SIDE Database Server может выполнять запрос примерно с такой же эффективностью, если поле **Customer No.** не является первым полем ключа. Например, если определен ключ, содержащий поле **Country/Region Code** в качестве первого поля ключа, и поле **Customer No.** как второе, а в записях используется мало различных стран/регионов, то запрос будет выполняться чуть дольше.

Ключи таблицы:

1. Entry No.
2. Country/Region Code, Customer No.,Posting Date

DRAFT

Запрос может выглядеть следующим образом:

```
SETCURRENTKEY("Country/Region Code","Customer No.");
SETRANGE("Customer No.",'1000');
IF FIND('-') THEN
  REPEAT
  UNTIL NEXT = 0;
```

Но SQL Server не сможет ответить на данный запрос эффективно, и поэтому будет выполнять чтение всей таблицы. В заключение, SQL Server предъявляет более строгие требования к тому, как ключи определены в таблицах и в запросах. При определении ключей нужно помнить об SQL Server, это гарантирует, что ваше приложение будет эффективно на обоих вариантах серверов.

30.5 Функции C/AL и производительность на SQL Server

Самый быстрый оператор SQL, который Dynamics NAV отправляет на SQL Server, выполняется медленнее, чем большинство функций базы данных на C/SIDE Database Server. Однако один оператор SQL может включать несколько обращений к серверу. В данном разделе описывается отношение между некоторыми основными функциями языка C/AL и операторами SQL.

Каждый GET (или FIND('=')) требует отдельного оператора SQL, если только клиент уже не получил запись в последней операции. Это означает, что если клиент несколько раз считывает запись, то обращение к SQL Server будет выполнено только, при первом считывании.

Каждый FIND('-/+') требует отдельного оператора SQL, если только клиент не выполнял такой же запрос в предыдущей операции.

Каждый NEXT (или FIND('>/<')) требует как минимум один, но чаще несколько операторов SQL.

Однако если NEXT используется вместе с FIND('-/+'), чтобы считать набор записей, как показано в следующем примере, то один оператор SQL может покрыть необходимость во всех вызовах функции NEXT, вызываемых в цикле:

```
IF FIND('-') THEN
  REPEAT
  UNTIL NEXT = 0;
```

Чтение в образном порядке с помощью FIND('+')/NEXT(-1) и с использованием "ASCENDING := FALSE" обладает одинаковой эффективностью. Не следует использовать для чтения записей "WHILE FIND('- /+') DO" или аналогичные конструкции.

Каждый CALCFIELD/CALCSUMS, который выполняет расчет суммы, требует отдельного оператора SQL, если только клиент не рассчитывал такую же сумму или другую сумму, использующую тот же SumIndex, фильтры и т.д. в последней операции. Другими словами, итоги рассчитываются для всех SumIndexFields в SumIndex, даже если требовалась сумма только по одному из них, все суммы хранятся в кэше клиента.

Каждый INSERT/MODIFY/DELETE требует отдельного оператора SQL. Если модифицируемая таблица содержит SumIndexes, операции будут значительно медленнее.

LOCKTABLE не требует отдельного оператора SQL.

DRAFT

Администрирование Базы Данных, Изменение Объектов и Производительность на SQL Server

Создание таблиц и фирм на SQL Server выполняется медленнее, чем на C/SIDE Database Server. Аналогично, перевод, переименование таблиц и полей медленнее на SQL

30.6 Хранимая Процедура Входа для SQL Server Option

Хранимая процедура входа это хранимая процедура, предназначенная для выполнения предопределенных функций после того как пользователь вошел на SQL Server Option. Типичной функцией является создание сообщения, информирующего пользователя, что база данные сейчас работает в режиме «single-user», чтобы администратор мог выполнить сервисные функции, поэтому в данный момент она не доступна.

Хранимая процедура входа выполняется немедленно после того как пользователь вошел на SQL Server и открыл базу данных, и перед тем как Dynamics NAV выполнит какие либо задачи, в том числе запуск C/AL триггеров. Пользователь должен успешно войти на сервер и иметь доступ к базе данных, перед тем как будет выполнена хранимая процедура.

Создание Хранимой Процедуры

Хранимая процедура создается в базе данных и имеет список параметров.

Хранимая процедура называется [ndodbproperty] и имеет следующие характеристики:

- Принимает два параметра типа VARCHAR: название приложения и номер версии C/SIDE. Эти параметры должны быть объявлены как часть хранимой процедуры, но не обязательны для использования.
- Может выполнять транзакции. Dynamics NAV использует COMMIT, чтобы очистить исходящие транзакции после того как хранимая процедура завершит выполнение.
- Оператор RAISERROR может использоваться для отображения сообщения об ошибке в Dynamics NAV и запрещает вход пользователя в базу данных.
- Оператор PRINT может использоваться для отображения предупреждающего сообщения Dynamics NAV и разрешает пользователю войти в базу данных.
- Если хранимая процедура возвращает значение, то оно игнорируется.
- Если хранимая процедура не существует, то Dynamics NAV не выполняет никаких действий и процесс входа выполняется как обычно.

Следующие примеры показывают, как создавать процедуры входа в Query Analyzer. Перед выполнением операторов база данных должна быть выбрана.

Пример 1

Отображает предупреждающее сообщение в Dynamics NAV и разрешает вход:

DRAFT

```
IF EXISTS (SELECT name FROM sysobjects
WHERE name = 'sp_$ndo$loginproc' AND type = 'P')
DROP PROCEDURE [sp_$ndo$loginproc]
GO
CREATE PROCEDURE [sp_$ndo$loginproc]
@appname VARCHAR(64) = NULL,
@appversion VARCHAR(16) = NULL
AS
BEGIN
PRINT 'The system will be unavailable on Sunday 1st April.'
END
GO
GRANT EXECUTE ON [sp_$ndo$loginproc] TO public
GO
```

Пример 2

Отображает сообщение об ошибке в Dynamics NAV и запрещает вход

```
IF EXISTS (SELECT name FROM sysobjects
WHERE name = 'sp_$ndo$loginproc' AND type = 'P')
DROP PROCEDURE [sp_$ndo$loginproc]
GO
CREATE PROCEDURE [sp_$ndo$loginproc]
@appname VARCHAR(64) = NULL,
@appversion VARCHAR(16) = NULL
AS
BEGIN
IF SUSER_SNAME() IN ('ACCOUNTS\jim', 'SALES\bill')
RAISERROR ('Please contact the system administrator.', 11, 1)
END
GO
GRANT EXECUTE ON [sp_$ndo$loginproc] TO public
GO
```

DRAFT