Объекты Navision

NAVISION®
Academy

Объекты Navision

Настоящий материал предназначен только для ознакомительных целей. Navision a/s не берет на себя никаких обязательств в случае использования его для иных целей. Navision a/s не несет ответственности за любые факты специального, прямого, косвенного или последующего ущерба, вне зависимости от того, порожден ли он выполнением контрактных обязательств, является следствием халатности, либо вытекает из использования или представления настоящего материала. Настоящий материал может подвергаться изменениям без какого-либо уведомления.

В соответствии с законодательством Дании об авторском праве, воспроизведение любой части данного материала в любой форме и любыми средствами допустимо только с разрешения Navision a/s.

Представленное в материале программное обеспечение поставляется на условиях лицензирования и должно использоваться и копироваться в строгом соответствии с положениями лицензионных соглашений.

COPYRIGHT

Copyright © 2002 Navision a/s, Frydenlunds Allé 6, 2950 Vedbaek, Denmark. All rights reserved.

ТОВАРНЫЕ ЗНАКИ

Товарные знаки, используемые в данном материале, и обозначаемые как ™ или ® являются соответственно товарными знаками или зарегистрированными товарными знаками Navision a/s или Navision Development a/s. Товарные знаки Microsoft, Windows, Windows NT, SQL Server и BackOffice являются зарегистрированными товарными знаками и товарными знаками корпорации Microsoft, действующими на территории США и других стран мира.

Товарные знаки Navision a/s и Navision Development представлены на сайте http://trademarks.navision.com

Использована гарнитура Arial.

Опубликовано Microsoft Business Solutions CIS.

DocID: AT-360-SST-005-v.01.00-RURU

Оглавление

Глава 1.

Введение **1-1**

1.1 О курсе 1-2
1.2 Основные объекты системы Navision 1-3
1.3 Основы среды C/SIDE 1-5

Глава 2.

Основные таблицы **2-1**

2.1 Что такое таблица 2-2
2.2 Свойства таблицы и поля 2-4
2.3 Создание таблиц 2-8
2.4 Что такое ключи 2-16
2.5 Установка отношений между таблицами 2-23
2.6 Специальные поля таблицы 2-34
2.7 Изменение некоторых свойств 2-50
2.8 Типы таблиц и их характеристики 2-56
2.9 Ответы на вопросы 2-72
2.10 Контрольные вопросы 2-81

Глава 3.

Основные формы **3-1**

3.1 Что такое формы 3-2
3.2 Работа с пустой формой 3-5
3.3 Другие полезные элементы управления 3-16
3.4 Использование мастера форм 3-29
3.5 Создание кнопок 3-37
3.6 Создание сопутствующих форм 3-45
3.7 Типы и характеристики форм 3-51

3.8 Стандартные средства навигации 3-59
3.9 Ответы на вопросы 3-62
3.10 Обзор главы 3-74

Глава 4.

Основные отчеты 4-1

4.1 Отчеты 4-2
4.2 Дизайнеры отчетов 4-7
4.3 Как работают разделы 4-18
4.4 Некоторые полезные свойства 4-24
4.5 Группировка в отчете 4-26
4.6 Суммирование в отчете 4-32
4.7 Добавление расширенных возможностей 4-36
4.8 Типы отчетов 4-42
4.9 Создание основного отчета 4-45
4.10 Ответы на вопросы 4-46
4.11 Обзор главы 4-55

Глава 5.

Основные порты данных 5-1

5.1 Основы порта данных 5-2
5.2 Конструирование портов данных 5-9
5.3 Примеры портов данных 5-17
5.4 Обзор главы 5-34

Глава 6.

Основные программные модули 6-1

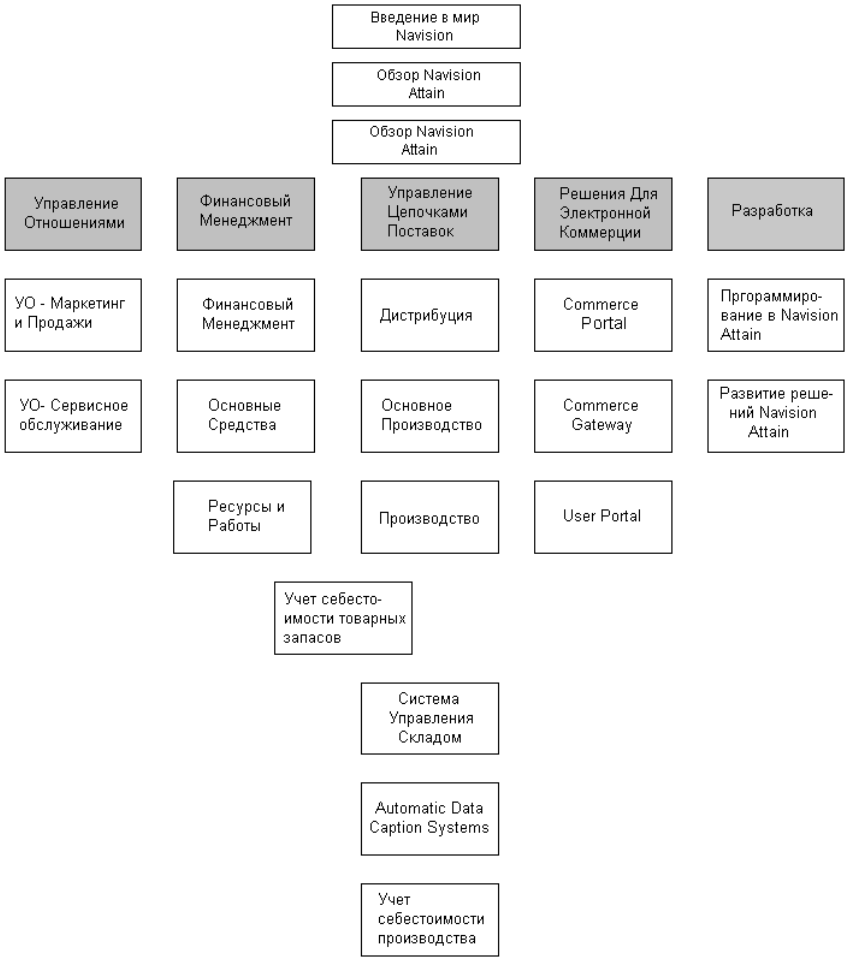
6.1 Программный модуль среды C/SIDE 6-2
6.2 Создание программных модулей 6-5
6.3 Использование программных модулей 6-13
6.4 Обзор главы 6-18

Предисловие

Настоящий курс является одним из четырех учебных курсов в рамках программы обучения “Программирование в Navision” (Navision Programming), которая, в свою очередь, входит в состав учебных материалов “Разработка решений Navision” (Navision Solution Development). Данный курс ориентирован на применение в качестве материала для самостоятельного обучения и может использоваться как отдельно, так и совместно с любым из трех других курсов в рамках данной учебной программы.

Когда вы завершите все четыре курса по программе “Программирование в Navision”, вы сможете пройти тестирование на соответствие требованиям сертификации в качестве консультанта в области программирования для Navision (Navision Attain Consultant in Programming).

На следующей диаграмме представлены все курсы в рамках программы обучения Navision (серым цветом выделены предметные области).



Глава 1

Введение

В настоящей главе вы познакомитесь с основными объектами системы Navision.

Глава содержит следующие разделы.

- О курсе;
- Основные объекты системы Navision;
- Основы среды C/SIDE.

1.1 О КУРСЕ

Для кого предназначен курс

Данный курс будет полезен всем, кто хотел бы познакомиться с основными объектами C/SIDE (Client/Server Integrated Development Environment – интегрированной среды разработки в клиент-серверной архитектуре) системы Navision. Перед использованием C/SIDE рекомендуется изучить данный курс.

Содержание курса

Благодаря курсу вы получите базовые сведения о создании и использовании основных объектов C/SIDE. Вы узнаете обо всех конструкторах объектов и поэкспериментируете с несколькими из десятков свойств каждого объекта. В курсе представлены следующие темы:

- Глава 1. Введение – данная глава
- Глава 2. Основные таблицы
- Глава 3. Основные формы
- Глава 4. Основные отчеты
- Глава 5. Основные порты данных
- Глава 6. Основные программные модули

Данный курс представляет собой самоучитель. На выполнение каждого упражнения вы можете затратить столько времени, сколько необходимо. Читайте материал как можно более внимательно. Если вам необходимо вернуться назад и повторить материал определенного раздела, постарайтесь найти для этого необходимое время. Для прохождения сертификации вы должны хорошо понимать пройденный материал.

1.2 Основные объекты системы Navision

В среде C/SIDE имеется пять основных объектов. Начнем с таблицы (table object). Таблица используется для определения того, как данные хранятся в базе данных и как их можно оттуда получать. Понимание природы таблиц существенно важно для использования других объектов, которые вам предстоит изучить.

Второй тип объекта, о котором вы узнаете, – это форма (form object). Форма используется для представления данных пользователю в привычном и удобном для него виде. Большинство форм позволяет пользователю добавлять записи в таблицу, а также просматривать и изменять их.

Затем вы изучите такой объект, как отчет (report object). Отчеты позволяют пользователю собирать и распечатывать подробные данные, используя фильтры и настраиваемую сортировку. Вы освоите проектирование и создание отчетов с использованием конструктора отчетов в среде C/SIDE.

Четвертый объект, с которым вы научитесь работать, – это порт данных (dataport object). Порты данных позволяют экспортировать и импортировать данные. Порты данных и отчеты основаны на единой архитектуре, поэтому здесь вы сможете применять те знания, которые получили при изучении отчетов.

И, наконец, последний тип объектов, с которым вам предстоит познакомиться, – это программный модуль (codeunit object). Программные модули позволяют вам как разработчику организовывать и группировать написанный вами код. Это единственный из упомянутых объектов, который, как видится на первый взгляд, не имеет никакого отношения к таблицам. На самом же деле и он связан с ними.

Является ли система Navision 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 среде C/SIDE вы располагаете всего пятью типами прикладных объектов. Вы можете создавать и использовать таблицы, формы, отчеты, порты данных, программные модули, и ничего более.

Это ограничение может показаться слишком жестким, но следует помнить, что при любом ограничении увеличивается свобода действий. Имея ограничения в виде этих пяти 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, что позволяет облегчить выполнение задач любому разработчику.

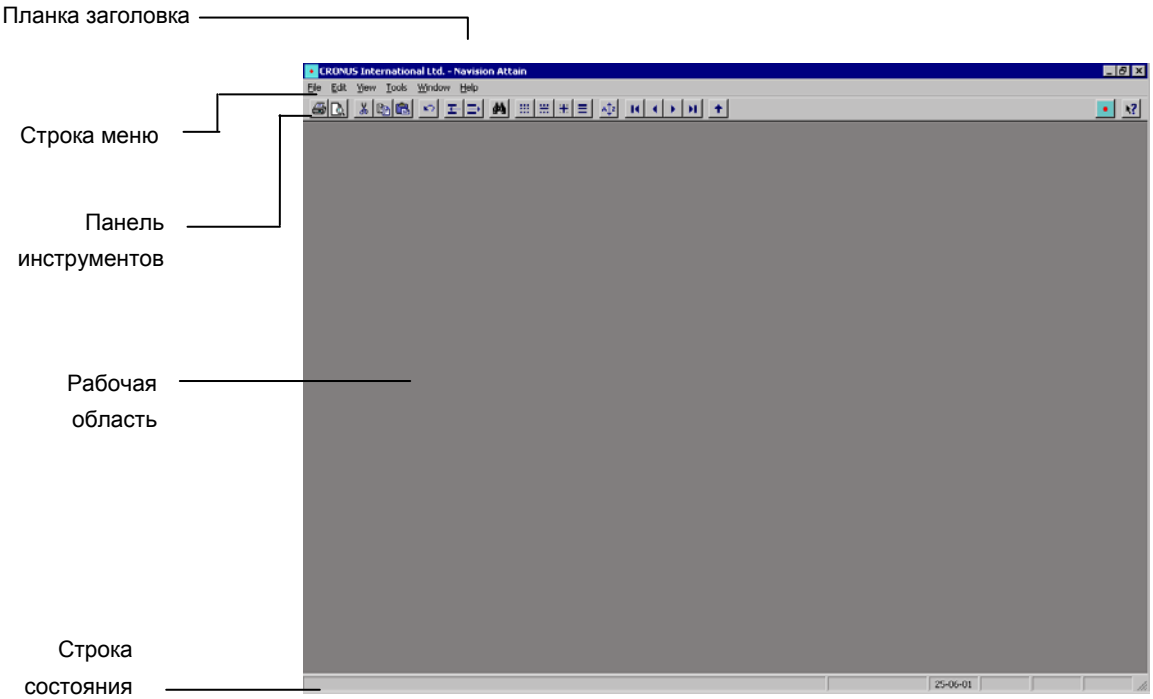
1.3 Основы среды C/SIDE

Приложение C/SIDE (Client/Server Integrated Development Environment – интегрированной среды разработки в клиент-серверной архитектуре) состоит из пяти типов прикладных объектов. Каждый тип прикладного объекта создан с использованием специального инструмента под названием “дизайнер”. Функционирование всех прикладных объектов, создаваемых вами 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.

Интерфейс C/SIDE

В данном подразделе вы познакомитесь с интерфейсом C/SIDE. Если вы еще не установили C/SIDE, обратитесь к руководству по установке. Если приложение C/SIDE уже установлено, программа установки создала новую группу, содержащую все значки, необходимые для работы в C/SIDE. Во время работы IDE (Integrated Development Environment – интегрированной среды разработки) экран монитора будет выглядеть так:



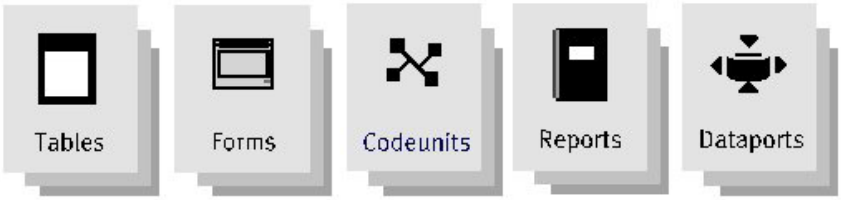
Интерфейс обеспечивает доступ к ряду инструментов и функций. Кроме того, некоторые части интерфейса предоставляют информацию о текущем состоянии системы. В приведенной ниже таблице дается объяснение того, когда использовать наиболее важные части интерфейса C/SIDE.

Чтобы…	**Используйте…**
получить информацию об имени текущей базы данных и пути к ней	планку заголовка
получить доступ к функциям в раскрывающемся меню	строку меню
быстро получить доступ к наиболее часто используемым функциям	панель инструментов (1)
работать с инструментами для конструирования приложения	рабочую область (2)
увидеть основные сведения о статусе системы (такие как текущая дата и ваш идентификатор пользователя)	строку состояния

1. Система автоматически меняет значки в зависимости от того, над выполнением какой задачи вы работаете.
2. Здесь пользователь также взаимодействует с вашими приложениями.

Конструирование прикладных объектов

Любое приложение, сконструированное в среде C/SIDE, основано на пяти различных типах прикладных объектов:



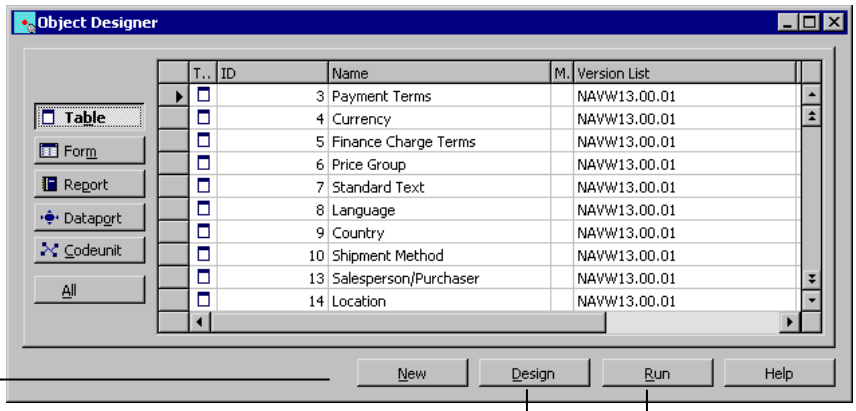
Таблицы – это основные объекты, в которых хранятся фактические данные. Для вставки данных в таблицы, а также для их изменения, удаления или показа нужны иные прикладные объекты. Обычно для ввода данных в БД и получения данных из БД используется форма, а для печати данных – отчет.

Примечание

Все прикладные объекты имеют идентификатор. Однако существуют ограничения на использование идентификаторов при создании прикладных объектов.

Основной инструмент, используемый для разработки приложений в среде C/SIDE – это Дизайнер объектов (Object Designer; выберите меню Tools, пункт Object Designer). Это инструмент, используемый для просмотра и конструирования таблиц, форм, отчетов, портов данных и программных модулей. В Дизайнере объектов вы выбираете тип прикладного объекта, над которым хотите работать. Чтобы модифицировать конструкцию существующего прикладного объекта или создать новый прикладной объект, из окна Дизайнера объектов вы можете запустить объект-приложение или дизайнер прикладного объекта. На следующем рисунке показано более подробно, как надо использовать Дизайнер объектов.

Здесь вы получаете доступ к дизайнерам для создания различных объектов. Вы просто выбираете тип объекта, над которым хотите работать



Создать новый объект

Изменить конструкцию текущего объекта

Открыть текущий объект

В приведенной ниже таблице перечислены инструменты, к которым можно получить доступ с помощью Дизайнера объектов, и условия, при которых их следует использовать.

Используйте…	**При работе над ...**
Дизайнер таблиц (Table Designer)	таблицами
Дизайнер форм (Form Designer)	формами
Дизайнер отчетов (Report Designer)	отчетами
Дизайнер портов данных (Dataport Designer)	портами данных
Редактор C/AL-кода (C/AL Editor)	программными модулями

Как видите, для каждого типа прикладного объекта существует определенный дизайнер. Создавая или модифицируя приложение, вы можете работать над любым количеством прикладных объектов одновременно, и каждый прикладной объект выводится на экран в собственном окне дизайнера. Например, если вы работаете одновременно над тремя формами, каждая форма может быть представлена в собственном окне Дизайнера форм. Единственный дизайнер, у которого не может быть более одного окна, – это Дизайнер объектов. При чтении остальных частей этой книги вы больше узнаете о том, как использовать каждый из этих дизайнеров.

Что такое приложение C/SIDE

Среда C/SIDE (Client/Server Integrated Development Environment – интегрированная среда разработки в клиент-серверной архитектуре) спроектирована 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

Вы уже узнали, что существует пять различных типов прикладных объектов в C/SIDE. Все пять типов 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**
Таблица	Таблица используется для хранение 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, триггеры
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, триггеры

Тип прикладного объекта	Цель использования	На каких концепциях основан тип
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, триггеры
Порт данных	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, триггеры
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, триггеры

Термины, содержащиеся в третьем столбце, описаны ниже.

Свойства. Свойства (properties) управляют внешним видом и поведением прикладных объектов, а также всех дочерних объектов (subobjects). 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. 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.

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.

Секции. Секция (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.

Физическая и логическая базы данных

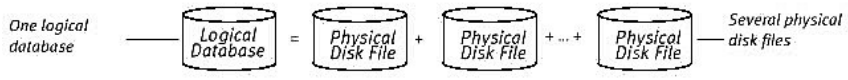
В предыдущем разделе описаны общие принципы, на которых основаны все пять типов прикладных объектов в среде C/SIDE. В этом разделе представлен другой взгляд на приложения C/SIDE. В соответствии с ним нас будет интересовать только то, как структурированы данные в вашем приложении.

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 данного курса 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 прозрачно для 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.



Логические структуры в вашей базе 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. (СУБД C/SIDE проводит различие между десятью различными типами информации.)

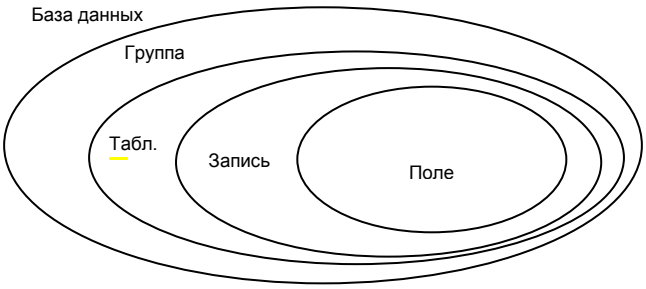
Поля объединяются в структуру под названием «запись» (record). Само по себе поле не обладает особенной значимостью, так как может содержать лишь ограниченный объем информации. Объединяя эти небольшие единицы информации в записи, мы получаем намного более гибкое «хранилище информации», которое к тому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

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 себе в виде матрицы размером N x M. Каждая строка N описывает запись, а каждый столбец M – поле записи. Таблицы объединяются в группы.

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

Группа – это наибольшая логическая структура, используемая в СУБД C/SIDE.



Глава 2

Основные таблицы

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. Упражнения, содержащиеся в данной главе, помогут укрепить знания, которое вы получите в этой части курса.

Глава содержит следующие разделы.

- Что такое таблица;
- Свойства таблицы и поля;
- Создание таблиц;
- Что такое ключи;
- Установка отношений между таблицами;
- Специальные поля таблицы;
- Изменение некоторых свойств;
- Типы таблиц и их характеристики;
- Ответы на вопросы;
- Контрольные вопросы.

2.1 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, содержащая девять строк и восемь столбцов. Каждая строка является записью, а каждый столбец – полем.

Столбец: поле

G/L Entry - Table

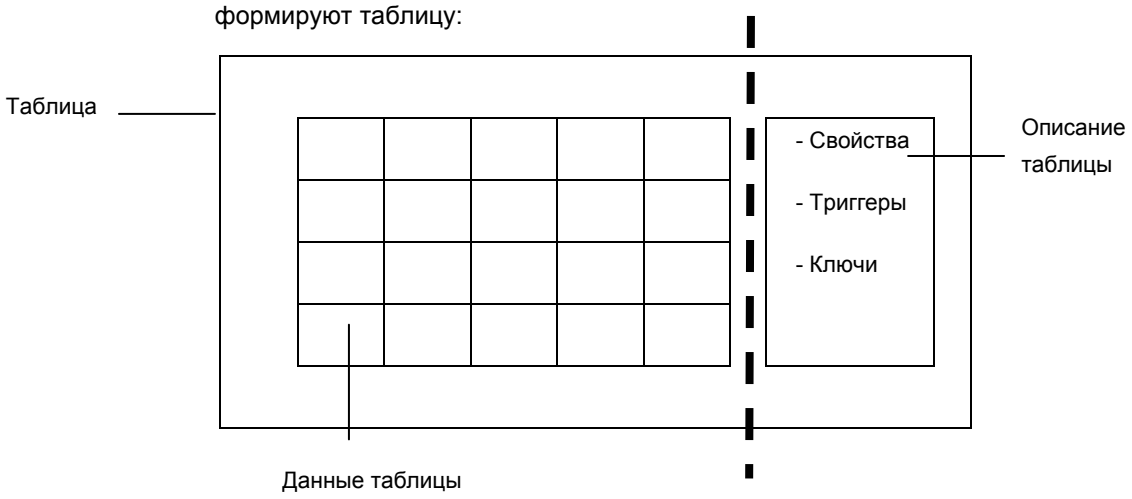
Entry No.	G/L Account	Date	Do Typ	Document No.	Description	Bal. Account	Amount
1	1110	C12/31/93		START	Opening Entry		11,344,104.
2	1140	C12/31/93		START	Opening Entry		-2,915,721.
3	1210	C12/31/93		START	Opening Entry		4,990,344.
4	1240	C12/31/93		START	Opening Entry		-3,101,574.
5	1110	C12/31/93		START	Opening Entry		1,322,682.
6	1140	C12/31/93		START	Opening Entry		-538,741.
7	1310	C12/31/93		START	Opening Entry		423,578.
8	1340	C12/31/93		START	Opening Entry		-212,361.
9	2120	C12/31/93		START	Opening Entry		2,311,205.

Help

Строка: запись

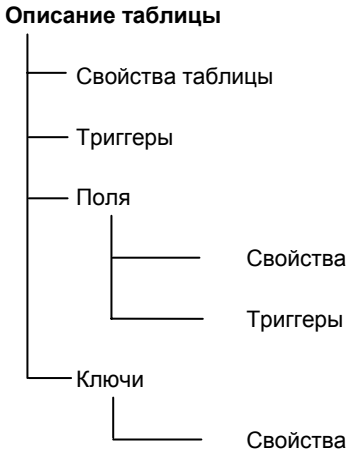
Таблица состоит из двух частей – данных таблицы и ее описания таблицы. Данные таблицы – та часть, которую пользователи часто считают содержимым базы данных, потому что оно состоит из записей со своими полями данных. Однако расположение и свойства этих полей определяются в описании таблицы.

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.

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. На рисунке также видно, что триггеры определены как для таблицы, так и для полей таблицы. Эти термины будут описаны более подробно в последующих разделах.

2.2 СВОЙСТВА ТАБЛИЦЫ И ПОЛЯ

Данный раздел содержит описание того, как используются свойства в структуре таблицы. В нем раскрывается то, как можно модифицировать и использовать свойства таблиц и полей.

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.

Для получения дополнительной информации о свойствах обратитесь к онлайновому «Справочному руководству» (Reference Guide).

Для просмотра или изменения свойств таблицы выполните следующие действия:

1 Нажмите кнопку Tools, Object Designer.

2 В окне Дизайнера объектов нажмите кнопку Table, чтобы получить список таблиц.

3 Выберите таблицу и нажмите кнопку Design. C/SIDE отобразит таблицу в окне Table Designer.

4 Установите курсор в пустой строке в Дизайнере таблиц или нажмите кнопку Edit, Select Object. (Если вы установите курсор в строке, определяющей одно из полей таблицы, то получите свойства данного поля, а не таблицы).

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

6 Если вы хотите изменить значение свойства, просто введите новое значение в окно **Properties**. Введя новое значение, обновите свойство, либо нажав клавишу ENTER, либо просто убрав курсор с поля.

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

Пример

Произведите описанные выше действия, чтобы увидеть свойство LookupFormID таблицы **Customer** (18). Это свойство сообщает C/SIDE, какую форму необходимо использовать для поиска значений в таблице **Customer**. Значение свойства – форма **Customer List**.

Просмотр и модификация свойств полей

Как и таблицы, все поля в C/SIDE имеют ряд 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.

Для получения дополнительной информации об указанных выше свойствах обратитесь к онлайновому Справочному руководству по C/SIDE.

Для просмотра или изменения свойств поля выполните следующие действия:

1 Нажмите кнопку Tools, Object Designer.

2 В окне Дизайнера объектов нажмите кнопку Table, чтобы получить список таблиц.

3 Выберите таблицу, и нажмите кнопку Design. C/SIDE отобразит таблицу в окне Дизайнера таблиц.

4 Установите курсор в строке Дизайнера таблиц, определяющей поле, для которого вы хотите получить доступ к свойствам.

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

6 Если вы хотите изменить значение свойства, просто введите новое значение в окне **Properties**. Введя новое значение, обновите свойство, либо нажав клавишу ENTER, либо просто убрав курсор с поля.

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

Пример

Произведите описанные выше действия, чтобы увидеть свойство NotBlank поля No.таблицы **Customer** (18). Это свойство не позволяет пользователю оставлять поле **No.** пустым. Оно используется в большинстве полей первичного ключа (вы узнаете о них позже).

Упражнение 2.2.1 Изучение существующих таблиц

В этом упражнении вам будут заданы вопросы о таблицах, которые уже существуют в основном приложении. Отвечайте на каждый вопрос как можно полнее.

Какова опциональная строка поля **Reserve** в таблице **Customer**?

Каково различие между формулами для вычислений поля **Balance** и поля **Balance At Date** в таблице **G/L Account** ?

Какое поле является полем **Date Filter** в таблице **G/L Account** ?

Каково значение свойства TableRelation поля **Currency Code** в таблице **Customer**?

2.3 Создание таблиц

Создание таблицы

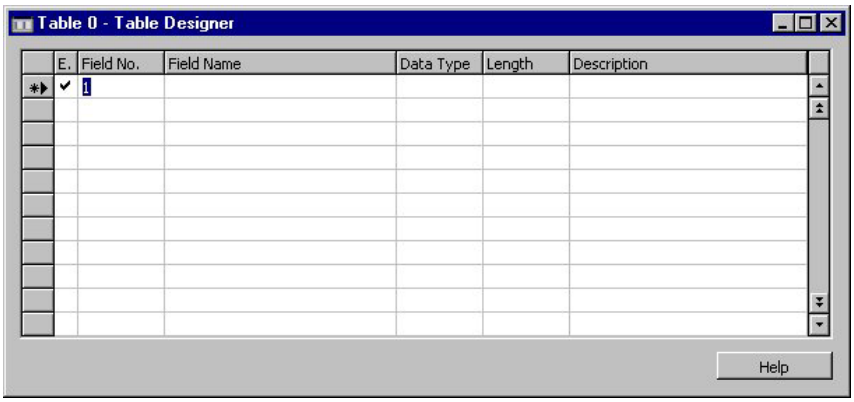
Новая таблица не содержит никаких данных. Создавая таблицу, вы должны решить, какие типы информации вы хотите в ней хранить.

Для создания таблицы выполните следующие действия.

1 Нажмите кнопку Tools, Object Designer. Появится окно **Object Designer**.

2 В окне **Object Designer** нажмите кнопку Table.

3 Нажмите кнопку 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. Как это сделать, описано в следующих подразделах.

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 типов данных

Выбрав идентификационный номер и имя поля, вы должны выбрать соответствующий тип данных. Вы можете использовать 11 различных типов 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 отчета 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 опции определяется строкой опции, 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.	4 байта

Тип данных	Описание	Размер
	Размер соответствующего типа данных SQL – INTEGER - 4 байта. (A)(B)	
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).
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 кода, равняется максимальной длине текста плюс два байта. Первый из этих дополнительных байтов содержит информацию о длине строки, а второй – о выравнивании. В Microsoft SQL Server Option for Navision Attain	Максимальная длина строки +2 байта (см. примечание ниже).

Тип данных	Описание	Размер
	поля кода функционируют по-другому. Вы можете использовать свойство SQL Data Type, чтобы указать, могут ли поля кода содержать целые числа или текстовые строки. Обратитесь к онлайновому «Справочному руководству по C/SIDE» (C/SIDE Reference Guide) для получения информации о свойстве SQL Data Type. Размер соответствующего типа данных SQL – VARCHAR – равен 1 байту на символ в значении поля. (A)(B)	
date	Значение даты в диапазоне от 1 января 0 г. до 31 декабря 9999 г. Неопределенная дата выражается 0. Все даты имеют соответствующий крайний срок. Система расценивает крайний срок для данной даты периодом, который следует за данной датой, но стоит перед следующей нормальной датой. Иначе говоря, крайний срок ставится сразу после соответствующей нормальной даты, но перед следующей нормальной датой. В Microsoft SQL Server Option дата не может быть меньше чем 01.01.1754. 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 (нуль) – 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	4 байта

Тип данных	Описание	Размер
	значениями «Yes» или «No». Размер соответствующего типа данных SQL – `TINYINT` - 1 байт. (A)(B)	
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).
BLOB	Binary Large Object – «большой двоичный объект». Используется для хранения растровых изображений и текстов. Обратите внимание, что BLOB хранится не в записи, а в области BLOB-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, когда это поле содержит формулу даты. Этот тип данных поддерживает многоязычность.	30 байт

(1) 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.

(2) Это тип данных SQL Server, который использует система Navision, создавая тип данных Navision. За более подробной информацией обращайтесь к *«Руководству разработчика приложений»* (*Application Designer's Guide).*

Примечание

В Navision Server данные хранятся с выравниванием на четыре байта 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.

Упражнение 2.3.1 Создание простой таблицы

В этом упражнении вы создадите простую таблицу.

1 В меню Tool выберите Object Designer

2 В окне Object Designer нажмите кнопку Table.

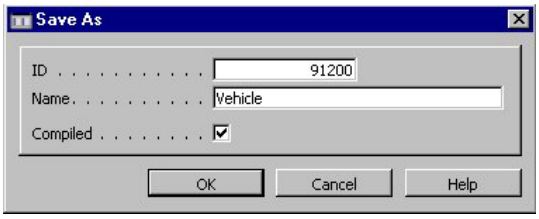
3 Нажмите кнопку New. C/SIDE отобразит окно Table Designer.

4 Введите Field Numbers, Field Names, Data Types и Lengths (если есть) для каждого поля.

№	Имя	Тип/длина
1	Модель (Model)	Code 20
2	Серийный № (Serial No.)	Integer
5	Описание (Description)	Text50
7	Трансмиссия (Transmission)	Option (автоматическая, 4-ступенчатая, 5-ступенчатая)
10	Цена по каталогу (List Price)	Decimal
15	Дата производства (Date of Mfg)	Date

5 Отредактируйте Field Properties, если необходимо.

6 Сохраните новую таблицу, закрыв окно Дизайнера, нажмите Yes в диалоге Save, а затем введите Table ID Number (91200) и Table Name (Vehicle) в диалоге Save As.



Упражнение 2.3.2 Добавление данных

В этом упражнении вы добавите данные к таблице, созданной в предыдущем упражнении.

1 В меню Tool выберите Дизайнер объектов.

2 В окне Object Designer нажмите кнопку Table.

3 Найдите таблицу 91200 и выберите ее.

4 Нажмите кнопку Run.

Теперь вы видите данные в новой таблице Vehicle. Обратите внимание, что в настоящий момент она пуста. Открытая форма была автоматически создана клиентским приложением, когда вы выбрали опцию запуска таблицы. Это основная табличная форма.

Теперь вы можете ввести некоторые данные для испытания своей таблицы и начать проверку ее функциональных возможностей.

5 Попытайтесь (обратите внимание на слово «попытайтесь») ввести следующие записи.

Модель	Серийный №	Описание	Трансмиссия	Цена по каталогу	Дата производства
FORD	5000	Красный, Mustang	Автоматическая	18000	01.15.00
FORD	2000	Синий,	5-ступенчатая	26000	01.15.00

Модель	Серийный №	Описание	Трансмиссия	Цена по каталогу	Дата производства
		F150			
TOYOTA	1000	Золотистый, Camry	Автоматическая	23000	02.01.01
FORD	3000	Черный, Explorer	Автоматическая	30000	01.15.00
TOYOTA	3000	Черный, Takoma	5-ступенчатая	20000	12.15.99
TOYOTA	2000	Серый, Camry	Автоматическая	22000	01.15.00

6 Ответьте на следующие вопросы:

С какой проблемой вы столкнулись?

Почему вы не можете ввести вторую запись?

7 Закройте форму. Давайте устраним обнаруженную проблему. Это будет сделано в следующем упражнении.

2.4 Что такое ключи

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).

База данных следит за каждой записью с помощью первичного ключа записи. Первичный ключ записи можно рассматривать в качестве имени этой записи. Имя каждой записи в определенной таблице для определенной компании должно быть уникальным. Это имя - то, как СУБД получает и обновляет 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.

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 в каждом поле в первичном ключе должно быть уникально не значение, а скорее комбинация значений во всех полях, из которых состоит первичный ключ.

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

Упражнение 2.4.1 Настройка первичного ключа

Изменение первичного ключа таблицы производится следующим образом.

1 В окне Object Designer нажмите кнопку Table.

2 Найдите таблицу 91200 и выберите ее.

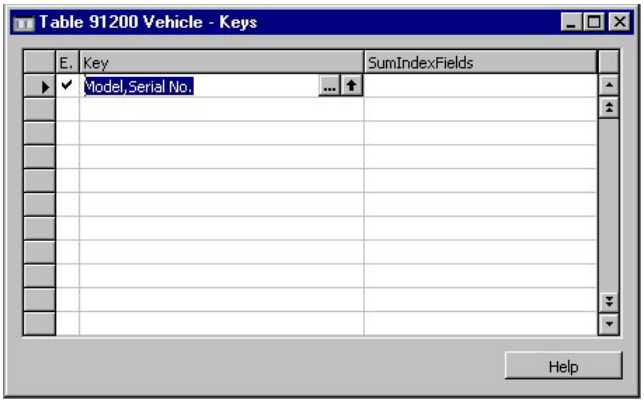
3 Нажмите кнопку Design. Откроется окно Table Designer для таблицы Vehicle .

4 В главном меню нажмите View, Keys. Будет отображено окно Key Designer. Обратите внимание, что первичный ключ содержит только одно поле – Model. Это означает, что вы не можете ввести две записи для одной и той же Model. В вашем примере это не лучший выбор первичного ключа. Если бы вы должны были включить Serial No. в основной ключ, вы бы могли вводить ту же самую модель много раз, пока поле Serial No. будет отличаться для каждого.

5 Щелкните в столбце Key на первой строке в окне Дизайнера ключей.

6 Нажмите кнопку Assist-Edit. Здесь вы можете выбрать другие поля, чтобы сделать их частью этого ключа.

7 Добавьте поле **Serial No.** к этому списку и нажмите кнопку OK.



8 Выйдите из окна Table Designer, сохраните и откомпилируйте таблицу. Первичный ключ изменен.

9 Закончите вводом данных из пункта 1. Обратите внимание, что записи сортируются в порядке полей первичного ключа (Model, затем Serial No.).

Vehicle - Table

Model	Serial No.	Description	T..	List Price	Date of ...
FORD	2000	Blue, F150	5..	26,000.00	01/15/00
FORD	3000	Black, Explorer	A..	30,000.00	01/15/00
FORD	5000	Red, Mustang	A..	18,000.00	01/15/00
TOYOTA	1000	Gold, Camry	A..	23,000.00	02/01/01
TOYOTA	2000	Gray, Camry	A..	22,000.00	01/15/00
TOYOTA	3000	Black, Takoma	5..	20,000.00	12/15/99

Help

Вторичные ключи

Мы уже упомянули, что до 40 ключей может быть связано с таблицей 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.

Примечание

Число полей в первичном ключе вместе со всеми полями в каждом вторичном ключе не должно превышать 20.

Другими словами, каждый вторичный ключ может содержать до 20 уникальных полей. Однако эти 20 уникальных полей должны также содержать все поля первичного ключа. Поля первичного ключа, которые не упомянуты специально во вторичном ключе, добавляются к концу ключа системой управления базами данных.

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.

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.

Упражнение 2.4.2 Создание вторичных ключей

Для некоторых ваших пользователей наиболее важная информация – это дата изготовления и тип трансмиссии. Эти пользователи хотели бы иметь возможность изменить порядок следования записей в форме, поставив на первое место Date of Mfg., Transmission. Чтобы сделать это, вы должны установить вторичный ключ.

1 В окне Object Designer нажмите кнопку Table.

2 Найдите таблицу 91200 и выберите ее.

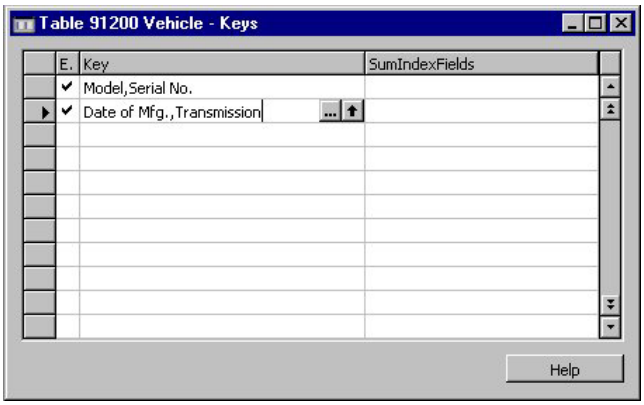
3 Нажмите кнопку Design. Откроется окно **Table Designer** для таблицы Vehicle .

4 В главном меню нажмите View, Keys. Будет отображено окно **Key Designer** .

5 Нажмите на пустую строку (это должна быть вторая строка).

6 Нажмите кнопку Assist-Edit. Здесь вы можете выбрать поля, чтобы сделать их частью этого ключа.

7 Выберите сначала поле **Date of Mfg.,** а затем - поле **Transmission**. Порядок полей важен. При описанном порядке записи будут сначала отсортированы по Date of Mfg., а затем – по Transmission (если некоторые даты совпадают).



8 Выйдите из окна **Table Designer**, сохраните и откомпилируйте таблицу. Вторичный ключ создан.

Упражнение 2.4.3 Использование вторичных ключей

В этом упражнении вы будете использовать вторичный ключ, созданный в упражнении 5.2.4.

1 В меню Tools выберите Object Designer.

2 В окне Object Designer нажмите кнопку Table.

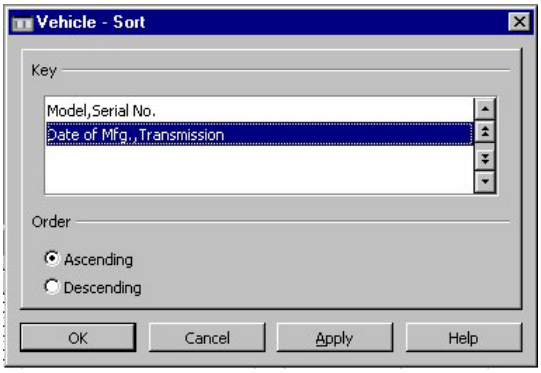
3 Найдите таблицу 91200 и выберите ее.

4 Нажмите кнопку Run.

5 Теперь вы видите данные в новой таблице Vehicle. Обратите внимание, что данные, которые вы предварительно ввели, по-прежнему находятся здесь. Кроме того, записи все еще сортируются в порядке, соответствующем первичному ключу.

Чтобы изменить сортировку записей в форме, выполните следующие действия.

1 Нажмите кнопку Sort на панели инструментов. Будет отображено окно Sort.



2 Выберите новый ключ, который вы создали (второй ключ в списке) и нажмите кнопку OK. Записи должны быть немедленно пересортированы с использованием нового ключа.

Vehicle - Table

Model	Serial No.	Description	T..	List Price	Date of ...
TOYOTA	3000	Black, Takoma	5..	20,000.00	12/15/99
FORD	3000	Black, Explorer	A..	30,000.00	01/15/00
FORD	5000	Red, Mustang	A..	18,000.00	01/15/00
TOYOTA	2000	Gray, Camry	A..	22,000.00	01/15/00
FORD	2000	Blue, F150	5..	26,000.00	01/15/00
TOYOTA	1000	Gold, Camry	A..	23,000.00	02/01/01

Help

Ответьте на следующие вопросы.

Посмотрите на записи, которые содержат Date of Mfg. от 01.15.00. В каком порядке расположены эти записи, если они содержат одну и ту же дату?

Посмотрите на записи, содержащие одни и те же Date of Mfg. и Transmission. В каком порядке расположены эти записи?

Что это говорит вам о вторичных ключах?

2.5 Установка отношений между таблицами

Если ваша база данных содержит таблицы со связанными данными, вы можете определить отношения между ними. Вы связываете таблицы, определяя одно или несколько полей, содержащих одинаковое значение в связанных записях. Эти соответствующие друг другу поля часто содержат одно и то же имя в каждой таблице. Вы можете использовать отношения для того, чтобы:

- проверять правильность ввода данных;
- производить поиск в других таблицах;
- автоматически переносить изменения из одной таблицы в другие.

Отношения таблиц и свойство TableRelation

Отношения таблиц определяются с помощью свойства TableRelation. Это свойство очень гибкое. Оно позволяет вам определять как простые, так и более сложные отношения таблиц. Типичное простое отношение таблиц состоит всего лишь из идентификатора таблицы и необязательного идентификатора поля, в то время как расширенные отношения таблиц обычно предваряются условным оператором и содержат фильтры.

Создавая отношения таблиц, вы определяете, какое поле искать в другой таблице. Если необязательный идентификатор поля оставлен пустым, это первое поле в первичном ключе, с которым установлено отношение. Чтобы установить отношение ко второму или третьему полю в первичном ключе, вы должны определить идентификатор поля.

Примечание

Идентификатор поля, указанный в свойстве TableRelation, должен находиться в первичном ключе таблицы, определенном идентификатором таблицы в свойстве.

Если указанное поле – не первое поле в первичном ключе, другие поля, стоящие перед ним в ключе, должны быть отфильтрованы для поиска одного значения. Это требование не определяется системой; оно имеет смысл только в конкретном приложении. Пользователя не должна смущать возможность выбора.

Упражнение 2.5.1 Простая таблица с отношениями таблиц

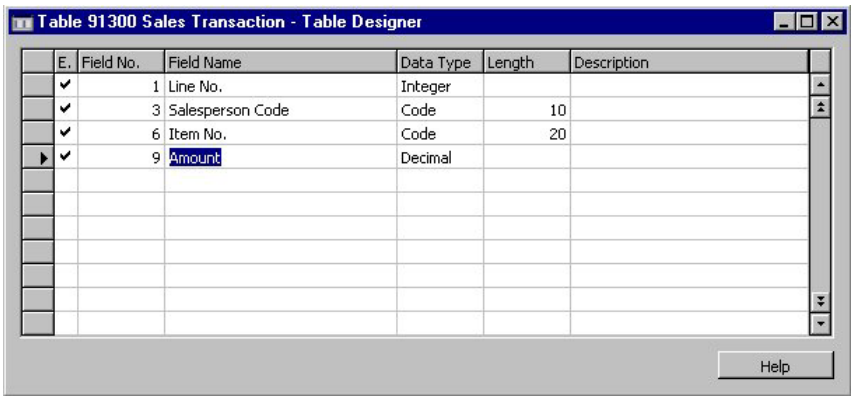
Создайте таблицу со следующими полями. Таблица будет содержать информацию о продаже товара. Пусть вас не беспокоит здесь вопрос о том, зачем вам понадобилась бы эта таблица. Она не имеет никакого конкретного применения, кроме достижения целей данного упражнения.

Первичный ключ	№ поля	Имя поля	Тип данных
Yes	1	Line No.	Integer
	3	Salesperson Code	Code 10
	6	Item No.	Code 20
	9	Amount	Decimal

Сохраните таблицу под именем **Sales Transaction** и с идентификатором 91300.

Теперь, когда основная таблица создана, добавьте два простых отношения таблиц.

1 Откройте таблицу **Sales Transaction** (91300) в Дизайнере объектов.



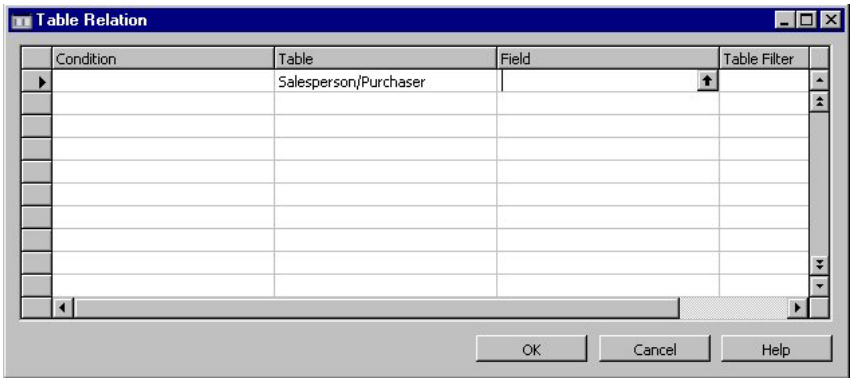
2 Выберите поле **Salesperson Code**. Это поле будет содержать значения из таблицы **Salesperson/Purchaser**. Пользователь может искать значение в таблице или вводить значение, но оно должно существовать в таблице **Salesperson/Purchaser**. Исходя

из этих требований, вы должны установить отношение таблиц в этом поле.

3 Откройте окно **Properties** поля **Salesperson Code** .

4 Выберите свойство TableRelation для поля **Salesperson Code** и нажмите кнопку поясняющего редактирования.

5 В окне **Table Relation** введите число *13,* идентификатор таблицы **Salesperson/Purchaser**, в поле **Table**. Вы можете также использовать кнопку поиска, чтобы выбрать таблицу или ввести имя таблицы.



6 Нажмите кнопку OK, чтобы сохранить изменения свойства.

7 Закройте, сохраните и откомпилируйте таблицу.

Теперь вам необходимо проверить отношение таблицы. Следующие действия покажут, насколько мощными являются отношения таблиц. Имейте в виду, что кнопки поиска на поле, которое имеет установленное отношение к таблице, открывает заданную по умолчанию форму поиска для связанной таблицы. Без формы для таблицы кнопка поиска не работает. В данном примере уже есть формы, определенные для таблицы **Salesperson/Purchaser**.

1 В окне Object Designer откройте таблицу 91300 (**Sales Transaction**).

2 В поле **Line No.** первой пустой строки введите *10000.*

3 Нажмите кнопку поиска в поле **Salesperson Code**, чтобы получить значение из таблицы **Salesperson/Purchaser** .

4 Выберите *AH* (Alice Hart) и нажмите кнопку OK. Обратите внимание, что поле **Salesperson Code** теперь имеет значение AH. Поскольку отношение TableRelation было установлено только к таблице (а не к какому-либо определенному полю), поиск возвращает значение из поля первого первичного ключа. Поскольку таблица **Salesperson/Purchaser** имеет только одно поле в первичном ключе, не было никакой необходимости в определении поля (в TableRelation можно выбирать только поля первичного ключа).

5 Удалите значение AH из поля **Salesperson Code** и введите только букву «a». Что произойдет, когда вы нажмете клавишу Enter?

6 Снова удалите значение и введите «XY». Что произойдет, когда вы нажмете клавишу Enter?

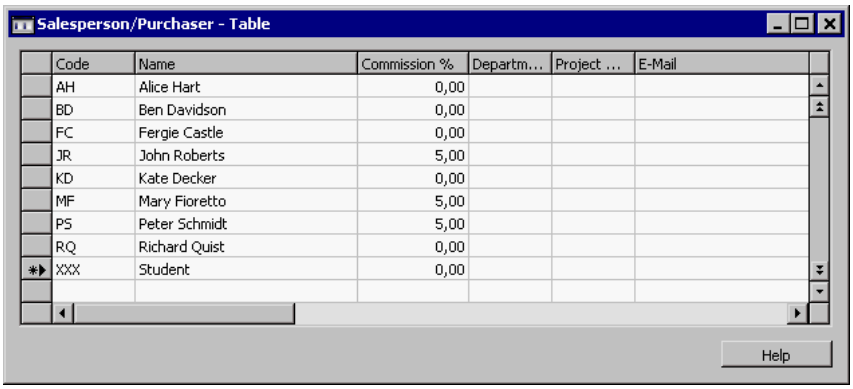
7 Закройте таблицу.

Упражнение 2.5.2 Переименование записей

Изменение значений первичных ключей записи в таблице приводит к изменению имени записи. Это называется переименованием записи. В этом упражнении вы увидите результаты переименования записи, которая связана с другими записями через отношение таблиц.

1 В окне Object Designer откройте таблицу 13 (**Salesperson/Purchaser**).

2 Выберите пустую строку и добавьте себя в качестве продавца. Введите свои инициалы в поле **Code** и свое имя в поле **Name**.



Code	Name	Commission %	Departm...	Project ...	E-Mail
AH	Alice Hart	0,00			
BD	Ben Davidson	0,00			
FC	Fergie Castle	0,00			
JR	John Roberts	5,00			
KD	Kate Decker	0,00			
MF	Mary Fioretto	5,00			
PS	Peter Schmidt	5,00			
RQ	Richard Quist	0,00			
XXX	Student	0,00			

3 Закройте таблицу 13.

4 Откройте таблицу 91300 (**Sales Transaction**).

5 Введите следующие строки (символы «XXX» обозначают ваши инициалы):

Строка №	Код продавца	№ товара	Количество
30000	XXX	70000	123.56
40000	AH	70000	234.97
50000	XXX	70010	90.99

6 Закройте таблицу 91300.

7 Откройте таблицу 13 (**Salesperson/Purchaser**).

8 Выберите строку, которую вы добавили ранее (запись с вашими инициалами и именем) и измените значение первичного ключа. Вместо инициалов введите первые три буквы своей фамилии.

9 Нажмите кнопку Yes, когда будет задан вопрос о том, хотите ли вы переименовать запись.

10 Сколько времени потребовалось, чтобы изменить это значение? Больше, чем вы ожидали?

11 Откройте таблицу 91300 (**Sales Transaction)** и проверьте данные, которые она содержит. Изменились ли какие-либо записи здесь? Почему?

12 Возможно, вы заметили, что не установили отношение таблиц в поле **Item No.**

Теперь вы должны связать поле **Item No.** с таблицей **Item** простым отношением таблиц.

Упражнение 2.5.3 Фильтрованные отношения таблиц

В этом упражнении вы добавите фильтры к одному из отношений таблиц, созданных в предыдущих упражнениях.

Возможно, вы заметили, что таблица 13 **(Salesperson/Purchaser)** содержит не только продавцов, но и покупателей. Вам нужно, чтобы пользователь выбрал в таблице только продавцов, а не покупателей, поэтому вы добавите фильтр к отношению таблиц. Продавцы имеют ставки комиссионного вознаграждения, в то время как покупатели – нет. Это позволит нам различать их.

1 Откройте таблицу 91300 в окне Дизайнера таблиц.

2 Выберите поле **Salesperson Code**. Это поле, которое необходимо изменить.

3 Откройте окно **Properties** для этого поля и выберите свойство TableRelation.

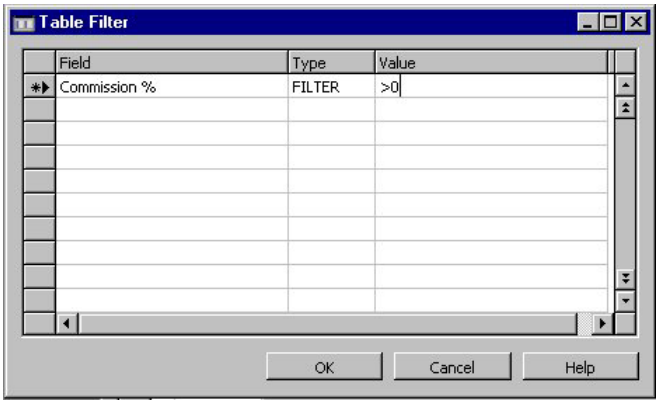
4 Нажмите кнопку поясняющего редактирования, чтобы открыть окно **Table Relation**. Обратите внимание, что заполнено только поле **Table**. Этому отношению вы также должны присвоить фильтр таблицы.

5 Выберите поле **Table Filter** и нажмите кнопку поясняющего редактирования, чтобы вывести на экран окно **Table Filter**. В этом окне вы можете определить фильтры на необходимом вам количестве полей в соответствующей таблице. Вы можете установить три различных фильтра для каждого поля, добавляемого к списку – постоянное значение, выражение для определения фильтра или поле в текущей таблице (91300). В данном случае вы будете использовать выражение фильтра.

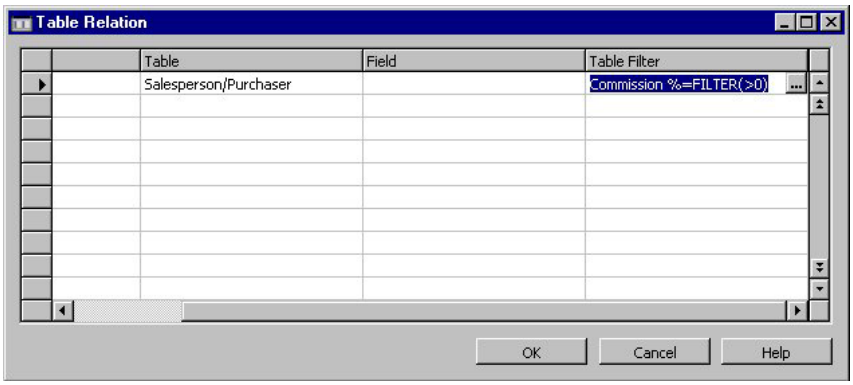
6 В первом столбце найдите и выберите поле **Commission %**. В нем хранится ставка комиссионного вознаграждения продавца.

7 Во втором столбце нажмите раскрывающуюся кнопку и выберите *Filter*. Вы должны выбрать фильтр исходя из значений, по которым хотите установить фильтр, а именно для случаев, когда Commission % больше нуля. С этой опцией фильтра вы можете использовать только операторы отношения. Варианты Const и Field – только точные соответствия (равно).

8 В третьем столбце введите *>0.*



9 Для сохранения фильтра таблицы нажмите кнопку OK.



10 Для сохранения отношения таблицы нажмите кнопку OK.

11 При открытом окне **Properties** переместите курсор мыши на поле **Value** свойства TableRelation . Это приведет к отображению всего отношения таблицы в виде текста. Убедитесь, что он содержит следующее (если данное упражнение не переделано):

```
Salesperson/Purchaser WHERE (Commission %=FILTER(>0))
```

12 Закройте окно **Properties** и сохраните, откомпилируйте и закройте таблицу.

13 Откройте таблицу 91300 и ответьте на следующие вопросы:

Когда вы используете кнопку поиска для поиска в поле **Salesperson Code**, все ли записи в таблице **Salesperson/Purchaser** вы видите? Если нет, то почему?

Можете ли вы ввести значение *AH* в поле **Salesperson Code**? Если нет, то почему?

Упражнение 2.5.4 Условные отношения таблиц

В этом упражнении вы добавите условия к одному из отношений таблиц, созданных в предыдущих упражнениях.

Пользуясь системой Navision, вы можете продавать несколько различных типов средств, включая товары, ресурсы и счета G/L. В своей таблице вы отслеживаете только товары. Было бы лучше иметь также номера ресурсов и номера счетов G/L в качестве составной части таблицы. Вы могли бы добавить еще два поля для хранения этих значений, но было бы более эффективно использовать одно уже имеющееся поле, которое могло бы хранить любой из этих трех типов. Чтобы добиться этого, необходимо добавить поле типа к таблице с тем, чтобы пользователь знал, является ли данный номер номером товара, номером ресурса или номером счета G/L .

1 Откройте таблицу 91300 в окне Дизайнера таблиц.

2 Добавьте в таблицу следующее поле. Обратите внимание, что используемый номер поля находится перед полем **Item No.** Это не критически важно, но, поскольку поле **Item No.** будет теперь зависеть от поля **Type**, поле **Type** следует заполнить первым.

Первичный ключ	№ поля	Имя поля	Тип данных
	5	Type	По выбору (Account,tem,Resource)

3 Для ввода выбранных параметров откройте окно **Properties** и введите параметры в поле **OptionString**. Вводите свойства очень внимательно. Они должны быть разделены запятыми и не должны содержать пробелов.

4 Теперь замените имя поля **Item No.** на имя **No.** Так будет надежнее, потому что этот товар мог не быть продан.

5 На этом этапе будет лучше, если вы сохраните, откомпилируете и закроете таблицу. После добавления нового поля к

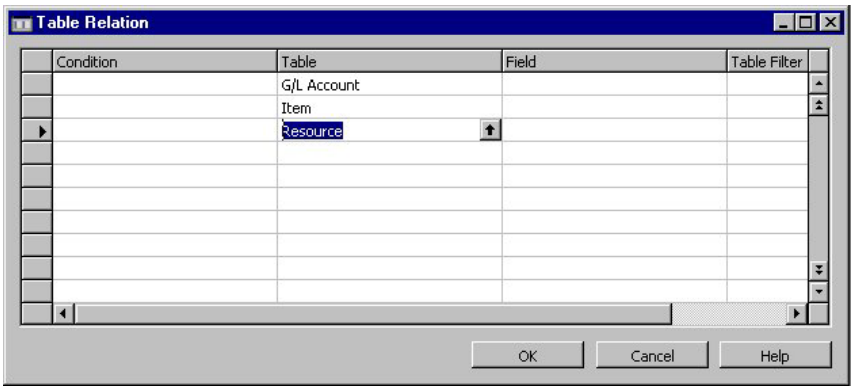
существующей таблице определение таблицы обновляется лишь после того, как вы сохраните таблицу. Благодаря этому действию выполнение последующих действий упрощается, потому что теперь новое поле будет отображаться в списке полей.

6 Снова откройте таблицу 91300 в Table Designer и выберите поле **No.** Необходимо изменить отношение таблицы для этого поля. Вы обеспечите возможность поиска в любой из следующих трех таблиц: **G/L Account**, **Item** или **Resource**.

7 Откройте окно **Properties** для поля **No.** и щелкните свойство **TableRelation**.

8 Нажмите кнопку поясняющего редактирования, чтобы открыть окно **Table Relation**.

9 Сначала введите имена этих трех таблиц в столбце Table. Поместите каждую таблицу в новой строке (возможно, сначала вам потребуется удалить предыдущую строку, которая была связана с таблицей Item). Однако вы не можете сохранить таким образом TableRelation. Необходимо установить условия. При наличии более чем одной строки в диалоге TableRelation вы должны установить условия.



10 Эти условия просты. Если поле **Type** – это Item (Товар), отношение должно быть установлено к таблице **Item**. И так далее. Отношения таблиц завершаются следующими значениями (мы произведем установку первого условия позже):

Условие	**Таблица**	**Поле**	**Фильтр таблицы**
Type=CONST(Account)	G/L Account		

Type=CONST(Item)	Item
Type=CONST(Resource)	Resource

11 В окне **Table Relation** выберите первую строку в столбце **Condition**.

12 Нажмите AssistButton, чтобы открыть окно **Table Filter**. Это то же самое окно, которые мы использовали для установки Table Filters для отношения таблицы. Однако в том, что касается условия, мы ограничены выбором CONST или FILTER (условие FIELD недоступно). См. приложение B для получения дополнительной информации.

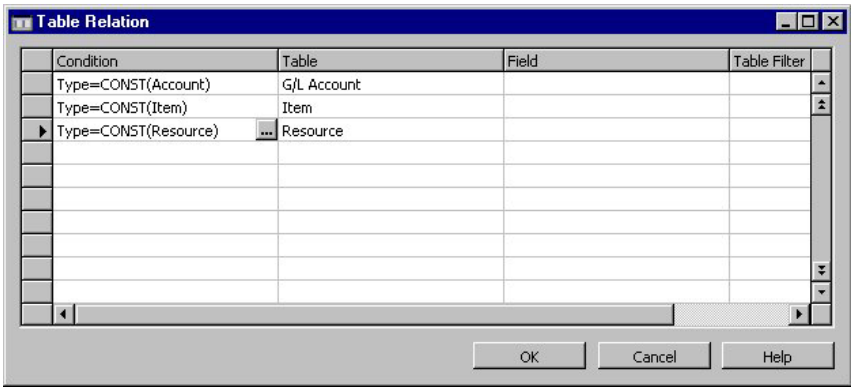
13 В столбце **Field** выберите поле **Type**.

14 В столбце **Type** выберите значение CONST (оно уже должно стоять по умолчанию).

15 В столбце **Value** введите *Account*. Кроме того, вы можете ввести цифру *0*, и после нажатия клавиши Enter или Tab нуль превратится в слово Account.

16 Нажмите кнопку OK, чтобы сохранить условие.

17 Повторите шаги 10-15 для других двух строк в окне **Table Relation**.



18 Нажмите кнопку OK в окне **Table Relation** чтобы сохранить свое условное отношение таблицы.

19 Закройте, сохраните и откомпилируйте таблицу.

Откройте таблицу 91300 и ответьте на следующие вопросы:

Когда вы используете кнопку поиска в поле **No.**, какая форма отображается? Что произойдет, если вы измените поле **Type**?

Замените поле **Type** на **Item** и попробуйте найти номер товара. Теперь замените поле **Type** на **Resource**. Что произошло? Является ли допустимым значение для поля **No.**?

Теперь заново введите значение в поле **No.** после замены **Type**, как описано в предыдущем вопросе. Какую ошибку вы получаете?

2.6 СПЕЦИАЛЬНЫЕ ПОЛЯ ТАБЛИЦЫ

В дополнение к обычным полям данных, которые просто хранят значения, для извлечения данных предназначены три типа специальных полей:

- SumIndexFields;
- FlowFields;
- FlowFilter Fields.

Что такое суммирующие поля SumIndexFields

SumIndexField – это десятичное поле, которое можно присоединить к определению ключа .

Это фундаментальная особенность баз данных в формате Navision, 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 записи базы данных.

Какие преимущества дают поля SumIndexFields

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

Например, вы хотите получить сумму всех значений в поле **Amount**. В обычной системе баз данных СУБД приходится получать доступ к каждой записи и добавлять каждое значение в поле **Amount**, что занимает очень много времени, если база данных содержит тысячи записей. Благодаря системе Navision для получения суммы 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 динамические поля FlowFields

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

Поле FlowFields и лежащая в его основе концепция суммирующих полей SumIndexFields разработаны для увеличения производительности в таких областях, как вычисление баланса клиентов, что в традиционных системах баз данных предполагает необходимость многократного получения доступа и производства множества вычислений, прежде чем будет получен результат. Как такой результат будет получен немедленно при использовании FlowFields, станет вам ясно при чтении остальной части этой главы.

Поле FlowFields не является постоянной частью данных таблицы. Динамическое 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 динамическом 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.

Существует семь типов динамических полей:

Тип FlowField	Тип поля	Описание
Sum	десятичное число	Сумма указанного набора в столбце таблицы.
Average	десятичное число	Среднее значение указанного набора в столбце таблицы.
Exist	булево значение	Указывает на то, существуют ли какие-либо записи в указанном наборе таблицы.
Count	целое число	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.

Формулы для производства вычислений и свойство CalcFormula

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

```
  Sum("Cust. Ledger Entry".Amount
  WHERE (Customer No.=FIELD(No.),
         Initial Entry Global Dim. 1 Code=FIELD(Global
Dimension 1 Filter),
         Initial Entry Global Dim. 2=FIELD(Global
Dimension 2 Filter),
         Posting Date=FIELD(Date Filter),
         Currency Code=FIELD(Currency Filter)))
```

Использование полей FlowFilter в формуле для производства вычислений

Конечному пользователю может понадобиться ограничить вычисления так, чтобы они включали только те значения столбца, которые имеют какие-то определенные свойства. Например пользователь может захотеть просуммировать только количество записей клиентов, введенных в апреле. Это возможно, если приложение было разработано с использованием полей FlowFilter во взаимосвязи с динамическими полями FlowFields.

Упражнения по реализации SIFT

В этом подразделе вы добавите вычисленное поле к таблице **Salesperson/Purchaser**, чтобы отобразить сумму связанных отчетов в таблице, которую вы создали в разделе 2.5. Для выполнения этого упражнения вы должны сначала справиться с упражнениями в разделе 2.5.

Упражнение 2.6.1 Вычисление сумм вручную

В этом упражнении вы вычислите суммы введенных вами данных. Эта информация поможет нам подтвердить суммы, которые вычисляются автоматически.

1 Откройте Дизайнер объектов и таблицу 91300.

2 Удалите все текущие записи из таблицы. Это позволит гарантировать, что суммы, которые вы получите, будут такими же, как в ответах, данных в конце настоящего руководства. Самый простой способ удаления всех записей - щелкнуть в левом верхнем сером прямоугольнике формы (это должно привести к выделению всех записей) и затем нажать клавишу F4 (это должно привести к появлению запроса о подтверждении и затем - удалению записей).

3 Теперь, когда таблица пуста, введите следующие записи в таблицу. Убедитесь в том, что используете предложенные значения.

Строка №	**Код продавца**	**Тип**	**№ товара**	**Количество**
10000	JR	Item	70000	300.00
20000	MF	Account	2910	150.00
30000	JR	Account	2920	200.00
40000	MF	Item	70000	100.00
50000	PS	Item	70000	110.00
60000	JR	Item	70010	50.00
70000	JR	Resource	LIFT	500.00
80000	MF	Resource	LIFT	550.00
90000	PS	Item	70010	75.00

4 Вычислите сумму для всей таблицы и для каждого уникального кода Salesperson Code. Запишите суммы в следующей таблице:

Какая сумма	Общее количество
For JR	
For MF	
For PS	
For ALL	

Обратите внимание, что это было бы намного проще, если бы вы сначала отсортировали записи по Salesperson Code. Конечно, единственный способ сделать это добавить ключ. Пока НЕ ДЕЛАЙТЕ этого.

Закройте таблицу.

Упражнение 2.6.2 Создание поля FlowField

В этом упражнении вы создадите поле FlowField, которое отображает сумму записей в таблице **Sales Transaction** (91300).

1 Откройте таблицу 13 **(Salesperson/Purchaser)** в окне Object Designer.

2 Добавьте в таблицу новое поле со следующими свойствами

Первичный ключ	№ поля	Имя поля	Тип данных
	91400	Sales	Десятичное число

Номер поля должен располагаться в пределах от 50 000 до 99 999, потому что таблица 13 – основная таблица приложения (ее номер меньше 50 000).

Тип данных должен быть десятичным, потому что вам нужно отобразить сумму (средние числа тоже предполагают наличие десятичного числа). Сумма FlowField вычисляет десятичное значение.

3 Откройте окно **Properties** для нового поля.

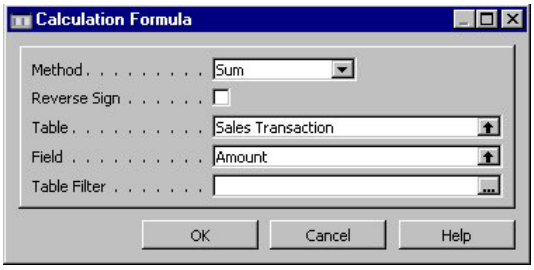
4 Перейдите в свойство Field Class и измените его на *FlowField* (поля FlowField используются для вычисления значения из другой таблицы).

5 Теперь доступно новое свойство с именем CalcFormula. Перейдите в это новое свойство.

6 Нажмите кнопку поясняющего редактирования, чтобы открыть окно **Calculation Formula**. Здесь вы сообщаете полю FlowField, как вычислить его значение.

7 Введите следующие значения в окно **Calculation Formula**:

Имя свойства	Значение
Method	Sum
Table	91300
Field	Amount



8 Нажмите кнопку OK, чтобы сохранить формулу для производства вычислений (нажатие x в диалоговом окне не приведет к сохранению изменений).

9 Закройте, сохраните и откомпилируйте таблицу. Если вы получили ошибку при попытке доступа, значит, вы ввели в поле неправильный идентификатор (см. пункт 2).

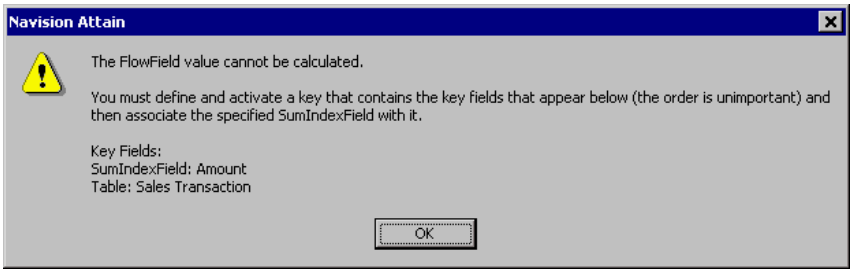
Сейчас вы успешно создали поле FlowField. Следующая задача – проверить его.

Упражнение 2.6.3 Тестирование поля FlowField

В этом упражнении вы протестируете созданное вами поле FlowField.

1 Откройте таблицу 13 (Salesperson/Purchaser).

Что случилось, когда вы попытались открыть таблицу?



Чем было вызвано сообщение об ошибке?

Что мы должны изменить, чтобы решить проблему (подсказка: очень внимательно прочитайте сообщение)?

Упражнение 2.6.4 Создание поля SumIndexField

Чтобы сумма вычислялась автоматически, вы должны создать SumIndexField. Это поле, связанное с ключом (индексом), который суммирует записи, сортируемые этим ключом (индексом). Чтобы ваше FlowField вычислило сумму, вы должны создать для него поле SumIndexField.

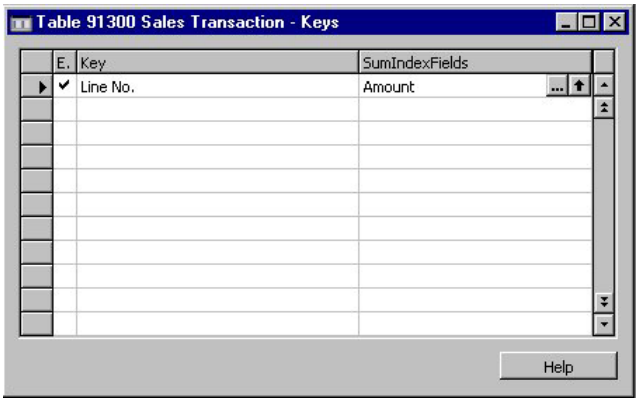
Вы можете просто узнать из сообщения об ошибке, что нужно делать. Через несколько мгновений вы узнаете о необходимой процедуре.

1 Откройте таблицу **Sales Transaction**, о которой упоминается в сообщении об ошибке в окне Table Designer.

2 Нажмите кнопку View, Keys , чтобы открыть окно Keys Designer. Здесь вы создадите ключ и поле SumIndexField.

3 Добавьте следующее поле SumIndexField к первичному ключу:

Включено	Ключ	SumIndexFields
x	Line No.	Amount



4 Закройте, сохраните и откомпилируйте таблицу.

Чтобы протестировать SumIndexField, вы должны просмотреть свое поле FlowField. Откройте таблицу 13 (**Salesperson/Purchaser**).

Были ли какие-либо ошибки на этой раз? Почему были или почему не были?

Заполните следующую таблицу значениями, отображенными в новом поле FlowField, Sales.

Какая сумма	Sales
Для JR	
Для MF	
Для PS	

Верна ли общая сумма? Какие есть подозрения на этот счет?

Соответствуют ли они какому-либо из значений из упражнения 5.4.1 (суммы, вычисленные вами вручную)? Каким значениям они соответствуют?

Что, как вам кажется, неверно в вашем поле FlowField?

Упражнение 2.6.5 Фильтрация поля FlowField

В этом упражнении вы откорректируете поле FlowField, с которым начали работать раньше. Проблема с FlowField заключалась в том, что вы не учли часть CalcFormula. Вы должны использовать Table Filter для большинства полей FlowFields. Фильтры таблиц позволяют пользователю суммировать не все, а лишь некоторые записи таблицы.

В поле FlowField, которое вы создали раньше, вам нужно суммировать только записи в таблице **Sales Transaction**, которая содержит определенный Salesperson Code. Например, если поле Code в таблице **Salesperson/Purchaser** - 'PS', то нам нужно лишь суммировать записи в таблице **Sales Transaction**, которая содержит Salesperson Code 'PS'.

Теперь используйте эту логику в свойстве CalcFormula поля FlowField.

1 Откройте таблицу 13 **(Salesperson/Purchaser)** в окне Table Designer.

2 Выберите поле **Sales**.

3 Откройте окно **Properties** и выберите свойство CalcFormula.

4 Нажмите кнопку поясняющего редактирования (assist-edit), чтобы открыть окно **Calculation Formula**.

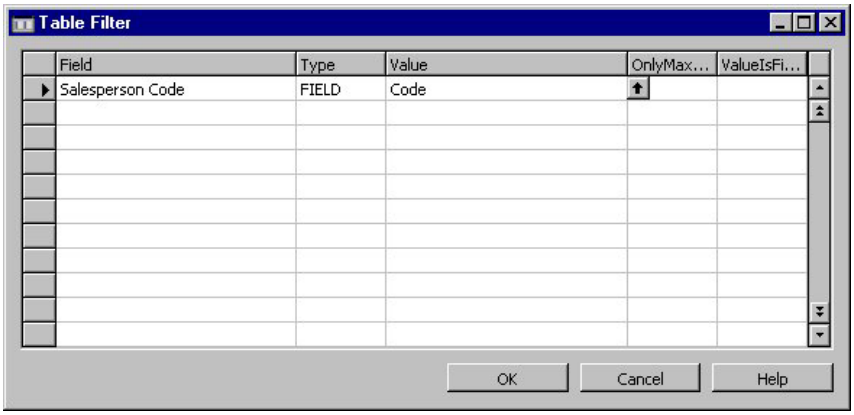
5 Обратите внимание, что в предыдущем упражнении вы оставили поле **Table Filter** пустым . Это часть свойства CalcFormula, которую необходимо заполнить. Когда вы оставляете его пустым, вычисление в поле FlowField основано на всех записях таблицы.

6 Нажмите кнопку поясняющего редактирования, чтобы открыть окно **Table Filter**. Установите следующий фильтр:

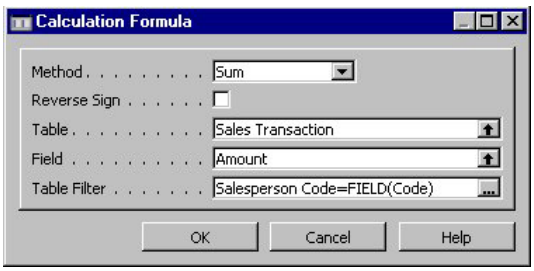
Поле	Тип	Значение	Ограничение OnlyMax	Фильтр значений
Salesperson Code	Field	Code		

Столбец **Field** соответствует полям таблицы **Sales Transaction** (таблица, данные которой вы суммируете). Поскольку в столбце **Type** вы установили значение Field **,** столбец **Value** представляет

поля в таблице, которую вы в настоящий момент проектируете **(Salesperson/Purchaser)**.



7 Для сохранения своего фильтра нажмите кнопку OK.



8 Для сохранения свойства CalcFormula нажмите кнопку OK.

9 Закройте, сохраните и откомпилируйте таблицу.

10 Откройте таблицу, чтобы посмотреть, дали ли сделанные вами изменения нужный результат.

Что произошло, когда вы попытались открыть таблицу?

Чем было вызвано сообщение об ошибке?

Что мы должны изменить, чтобы решить проблему (подсказка: очень внимательно прочитайте сообщение)?

Упражнение 2.6.6. Процесс создания поля SumIndexField

В этом упражнении вы изучите надлежащую процедуру принятия решения для создания поля SumIndexFields.

Первое, что необходимо сделать, – это посмотреть на свойство CalcFormula поля FlowField. Это поможет вам понять, как создать поле SumIndexField.

1 Откройте таблицу 13 **(Salesperson/Purchaser)** в окне Table Designer.

2 Выберите поле **Sales**.

3 Откройте окно **Properties** и выберите свойство CalcFormula.

4 Нажмите кнопку assist-edit, чтобы открыть окно **Calculation Formula**.

5 Запишите имя таблицы, указанное в окне ниже.

Имя таблицы: _____

Это таблица, в которой должно быть создано поле SumIndexField.

6 Запишите имя поля, указанное в окне ниже.

Имя поля: _____

Это поле, которое должно быть присоединено к ключу для создания необходимого вам поля SumIndexField. Это поле должно содержать десятичные числа и находиться в таблице, которую вы записали в пункте 5.

7 Нажмите кнопку поясняющего редактирования в поле Table Filter, чтобы открыть диалоговое окно Table Filter. Запишите все поля, упомянутые в столбце Field (первый столбец). (Вы можете использовать не все эти промежутки).

Имя поля 1: _____

Имя поля 2: _____

Имя поля 3: _____

Это поля таблицы (данные которой вы суммируете), согласно которым вы должны произвести сортировку. Вам необходим ключ, содержащий все эти поля. Порядок полей в ключе не имеет значения для данного поля FlowField, хотя вы можете использовать этот ключ и для других целей.

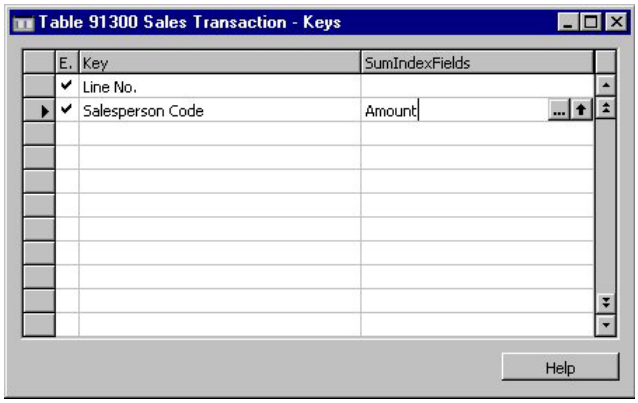
8 Теперь у вас есть таблица, поле и ключ. Закройте эту таблицу и спроектируйте таблицу, которую вы определили в пункте 5.

9 Нажмите кнопку View, Keys , чтобы открыть окно Keys Designer. Здесь вы создадите ключ и поле SumIndexField.

10 Создайте новый вторичный ключ, имеющий следующие свойства:

Включено	Ключ	SumIndexFields
x	Salesperson Code	Amount

Если это совпадает с тем, что вы написали при выполнении пунктов 5-7, начните с выполнения пункта 1.



11 Закройте, сохраните и откомпилируйте таблицу.

Чтобы протестировать поле SumIndexField, вы должны просмотреть поле FlowField. Откройте таблицу 13 (Salesperson/Purchaser).

Были ли какие-либо ошибки на этой раз? Почему были или почему не были?

Упражнение 2.6.4 Создание FlowFilter

В этом упражнении вы расширите функциональные возможности поля FlowField, включив в него фильтр FlowFilter. FlowFilter позволяет пользователю динамически изменять такую часть поля FlowField как Table Filter .

Вы создадите динамический фильтр FlowFilter, который позволит вам как пользователю изменять сумму на основе поля Type таблицы Sales Transaction. Это даст возможность пользователю применить ваш фильтр FlowField для подсчета четырех различных сумм – суммы за счета продаж, суммы за проданные товары, суммы за проданные ресурсы или суммы всех продаж.

Вы начинаете с изменения основной таблицы (таблица 13, **Salesperson/Purchaser).**

1 Откройте таблицу **Salesperson/Purchaser** в окне Table Designer.

2 Создайте новое поле, имеющее следующие свойства:

Первичный ключ	№ поля	Имя поля	Тип данных
	91401	Type Filter	По выбору (Account,Item,Resource)

Помните, что параметры должны быть введены именно так, как они представлены в этой таблице.

3 Выберите новое поле и откройте окно **Properties**.

4 Замените свойство Field Class на *FlowFilter.* Теперь это - поле FlowFilter, но вы еще не закончили выполнение задачи. Если бы сейчас надо было открыть таблицу, вы могли бы заменить поле **Type Filter**, но ничего бы не произошло. Вам нужно сделать так, чтобы поле FlowField использовало FlowFilter.

5 Закройте, сохраните и откомпилируйте таблицу. Это действие позволит упростить выполнение нескольких следующих шагов.

6 Снова откройте таблицу 13 в окне Table Designer.

7 Выберите поле **Sales**.

8 Откройте окно **Properties** и выберите свойство CalcFormula.

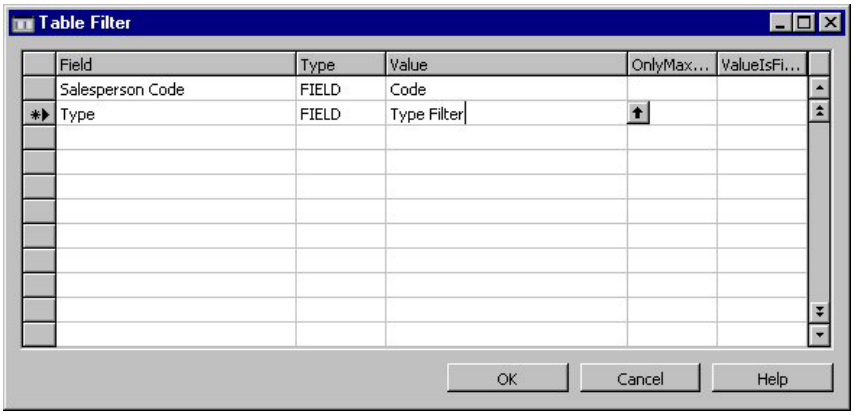
9 Нажмите кнопку поясняющего редактирования, чтобы открыть окно **Calculation Formula**.

10 В поле **Table Filter** нажмите кнопку поясняющего редактирования, чтобы открыть окно **Table Filter**. Здесь вы должны изменить поле FlowField так, чтобы оно использовало FlowFilter.

11 Щелкните в пустой строке (это должна быть вторая строка в диалоговом окне). В эту пустую строку введите следующее:

Поле	**Тип**	**Значение**	**Ограничение OnlyMax**	**Фильтр значений**
Type	Field	Type Filter		

Помните, что столбец Field соответствует полям таблицы **Sales Transaction**. Поскольку в столбце **Type** вы установили значение Field**,** столбец **Value** представляет поля в таблице, которую вы в настоящий момент проектируете **(Salesperson/Purchaser)**.



12 Для сохранения своего фильтра нажмите кнопку OK.

13 Для сохранения свойства Calculation Formula нажмите кнопку OK.

14 Закройте, сохраните и откомпилируйте таблицу.

15 Откройте таблицу, чтобы посмотреть, дали ли сделанные вами изменения нужный результат.

Вы можете не заметить различий, но это нормально. Это означает, что ничего не нарушено. В следующем упражнении вы протестируете непосредственно FlowFilter .

Упражнение 2.6.8 Использование FlowFilter

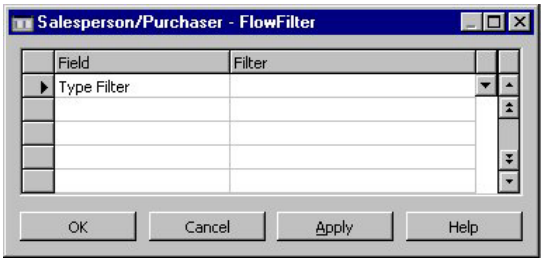
В этом упражнении вы научитесь использовать FlowFilter как пользователь. Кроме того, вы узнаете, что изменения, вносимые вами в FlowFilter, могут привести к неожиданным результатам.

1 Откройте таблицу 13 (**Salesperson/Purchaser**).

2 Тщательно проверьте значения в столбце Sales.

3 Вы можете заметить, что поле **Type Filter** не присутствует на нашей форме. Это следствие того, что FlowFilters имеет специальную форму. Нажмите кнопку FlowFilter на панели инструментов или нажмите View, FlowFilter, чтобы отобразить эту специальную форму. Это окно FlowFilter. Здесь вы можете изменить значение FlowFilter.

   Помните, что изменение значения FlowFilter должно привести к изменению значения FlowField.



4 В столбце Filter поля **Type Filter** выберите тип Item. Пока ничего не произошло. Вам еще нужно нажать кнопку OK или Apply.

5 Нажмите кнопку OK.

Что произошло?

Почему вы получили сообщение только после того, как установили FlowFilter?

Какое сообщение об ошибке вы получили?

Вы видели раньше подобное сообщение?

Какое действие вы должны предпринять дальше, чтобы решить проблему?

Упражнение 2.6.9 Решение проблемы.

1 Откройте таблицу **Sales Transaction** в окне Table Designer.

2 Нажмите кнопку View, Keys.

3 Добавьте поле **Type** во вторичный ключ, который вы создали ранее, так, чтобы ключ выглядел следующим образом:

Включено	Ключ	SumIndexFields
x	Salesperson Code,Type	Amount

Вы могли бы создать новый ключ, но в этом нет необходимости. Этот ключ будет работать для поля FlowField независимо от того, используете вы фильтр FlowFilter или нет.

Чтобы проверить, исправлена ли ошибка, повторите действия 3-5 из предыдущего упражнения.

Все ли работало правильно тогда?

2.7 Изменение некоторых свойств

В этом разделе вы изучите некоторые очень полезные свойства и узнаете, почему следует их использовать.

Упражнение 2.7.1 Свойство NotBlank

Благодаря этому упражнению вы поймете необходимость свойства NotBlank. Вы начнете с ввода некоторых записей.

1 Откройте таблицу 91100 (**Vehicle**).

2 Нажмите клавишу F3, чтобы вставить новую запись.

3 Не вводя значения в поля **Model** или **Serial No.**, щелкните поле **Description**.

4 Введите *New Vehicle* в качестве описания.

5 Щелкните другую запись в таблице (неважно, какую). Обратите внимание, что ваша запись «New Vehicle» вставлена в таблицу.

Какое «имя» имеет эта запись (помните, что имя – это значение первичных полей ключа)?

Что произойдет, если вы повторите действия 2-5, описанные выше?

Как мы могли бы удалить пустые записи, помещаемые в таблицу случайно?

6 Откройте таблицу 91100 **(Vehicle)** в окне Table Designer.

7 Откройте окно **Properties** для поля **Model**.

8 Измените значение свойства NotBlank на *Yes*.

9 Закройте, сохраните и откомпилируйте таблицу.

10 Откройте таблицу.

11 Повторите действия 2-5.

Что происходит на этот раз?

Что вы должны сделать теперь, чтобы вставить новую запись?

Примечание

Если вы попытаетесь использовать свойство NotBlank в полях, которые не содержатся в первичном ключе, оно не будет вести себя таким же образом. Повторите описанные выше действия по отношению к полю **Description** и посмотрите, работает ли NotBlank таким же образом в нем.

Упражнение 2.7.2 Свойство Editable

Благодаря этому упражнению вы поймете необходимость свойства Editable. Вы начнете с изменения некоторых записей.

1 Откройте таблицу 13 (**Salesperson/Purchaser**).

2 Щелкните столбец Sales. Обратите внимание, что это поле доступно для редактирования (может быть изменено). Помните, что вы создали поле FlowField. Поля FlowField – это вычисленные значения.

3 Щелкните поле **Sales**, чтобы отобразить продавца PS.

4 Введите число *500* как новое значение Sales.

Что произошло?

Сохранилось ли изменение (если вы закрыли таблицу и снова открыли ее, значение PS по-прежнему 500)?

Откройте таблицу 91300 (**Sales Transaction**) и вручную вычислите сумму значений PS (возможно, сначала вам потребуется изменить порядок сортировки)

Какова новая сумма? Какой был старая сумма (раздел 2.7.1)? В чем различие?

Можете ли вы определить, какая строка привела к изменению суммы?

Вы ввели эту строку? Как она была введена?

Эту ситуацию нельзя назвать хорошей. Пользователям, видящим поле FlowField, нельзя позволять случайно изменять сумму. Пользователь немедленно получил бы ошибку, если бы попытался изменить другие типы динамических полей FlowFields (среднее число, существующее и так далее). Поэтому стоит взять за правило делать большинство динамических полей FlowFields недоступными для редактирования.

Чтобы сделать поле (FlowField или любое другое) недоступным для редактирования, требуется предпринять следующие действия:

1 Откройте таблицу 13 в окне Table Designer.

2 Щелкните поле, которое вы хотите измениться (поле **Sales**).

3 Откройте окно **Properties**.

4 Щелкните свойство Editable (НЕ свойство Enabled).

5 Измените значение на *No*.

6 Закройте, сохраните и откомпилируйте таблицу.

7 Проверьте изменение, открыв таблицу и попытавшись отредактировать поле.

Примечание

В некоторых случаях поведение, которое вы наблюдали, – именно то, чего вы хотите добиться. Эти функциональные возможности используются в G/L Budgets, например, чтобы упростить пользователю задачу ввода информации о бюджете. Вам стоило бы посмотреть на бюджеты, чтобы понять, как это делается.

Упражнение 2.7.3 Свойство DecimalPlaces

В этом упражнении вы измените свойство DecimalPlaces поля, чтобы обеспечить большую точность.

1 Откройте таблицу 91300 (**Sales Transaction**).

2 Введите новую запись со следующими значениями (обратите внимание, что количество содержит три десятичных разряда).

Строка №	**Код продавца**	**Тип**	**№ товара**	**Количество**
130000	JR	Item	70000	301.155

Удалось ли вам правильно ввести количество?

Какую ошибку вы получили?

По умолчанию свойство DecimalPlaces – 2:2. Цифра перед двоеточием – это минимальное число десятичных чисел, которые будут отображены для пользователя. Цифра после двоеточия – максимальное число десятичных чисел, которые может вводить пользователь .

В данном случае вы не можете ввести три десятичных разряда, потому что после двоеточия стоит цифра два. Чтобы изменить это положение, выполните следующие действия.

1 Откройте таблицу 91300 (**Sales Transaction**) в окне Table Designer.

2 Откройте окно **Properties** для поля **Amount**.

3 Щелкните свойство DecimalPlaces.

4 Введите значение *2:3.*

5 Закройте, сохраните и откомпилируйте таблицу.

6 Откройте таблицу.

7 Теперь завершите ввод количества и нажмите клавишу Enter.

Что произошло?

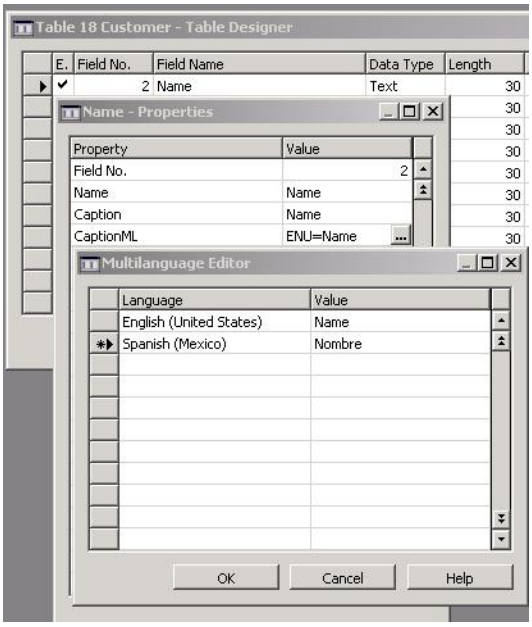
Что отображается в поле, если вы вводите количество 234.1 и нажимаете клавишу Enter?

Что отображается в поле, если вы вводите количество 234 и нажимаете клавишу Enter?

Что отображается в поле, если вы вводите количество *234,110* и нажимаете клавишу Enter?

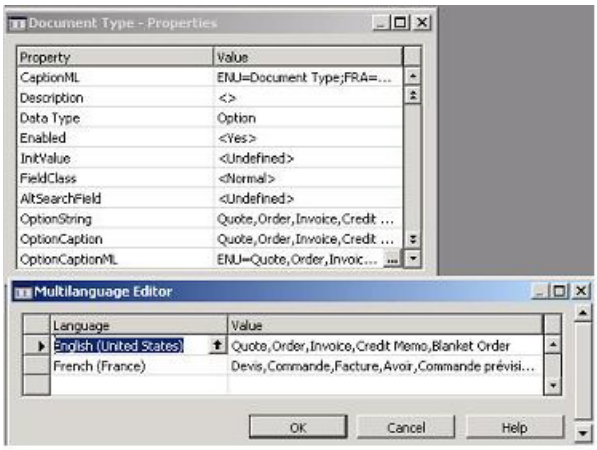
Упражнение 2.7.4 Свойства CaptionML и OptionCaptionML

Все объекты и поля имеют свойство CaptionML. Это свойство позволяет пользователю переключать языки и обеспечивать отображение заголовков на выбранном языке. Можно добавить дополнительные языки, и для каждого языка значение свойства CaptionML установлено на нужный термин языка. Английский язык (Соединенные Штаты) (ENU) обязателен. При создании новой таблицы перейдите в свойства Table, чтобы ввести имя таблицы для CaptionML. См. пример для поля **Name** CaptionML ниже.



Свойство OptionCaptionML для Option Data Types тоже должно быть заполнено.

На следующем рисунке изображено окно **Properties** для поля **Document Type**.



2.8 Типы таблиц и их характеристики

В системе Navision существуют множество стандартных типов таблиц. В следующей таблица перечислены наиболее общие типы таблиц и приведены некоторые примеры каждой из них. Более подробное описание каждого типа дано в следующих подразделах.

Понимание стандартных типов таблиц системы Navision поможет вам быстро понять структуру различных областей приложения.

Тип	Краткое описание	Примеры
Master	Содержит информацию об основном предмете ее функциональной сферы.	Customer, Vendor, Item
Supplemental	Хранит информацию о дополнительном предмете, используемом в одной или нескольких функциональных сферах.	Language, Currency
Setup	Хранит одну запись, которая содержит общую информацию о ее функциональной сфере.	G/L Setup, Sales &Receivables Setup
Register	«Содержание» соответствующей таблицы Ledger или таблиц.	G/L Register, Item Register
Subsidiary	Содержит информацию, являющуюся дополнительной либо для таблицы Master, либо для таблицы Supplemental, либо для них обеих.	Item Vendor, FA Depreciation Book
Ledger	Содержит информацию о сделках, являющуюся основным предметом ее функциональной сферы.	Cust. Ledger Entry, Item Ledger Entry
Journal	Таблица для ввода основной информации о сделках для функциональной сферы.	Purchase Journal, Item Journal

Тип	Краткое описание	Примеры
Document	Таблица для ввода информации о второстепенных сделках, которые допускают ввод в функциональную сферу или сразу в несколько функциональных сфер. Она фактически реализована как пара таблиц.	Sales Header / Sales Line, Finance Charge Memo Header / Finance Charge Memo Line, Reminder Header / Reminder Line
Document History	Содержит историю операций для оприходованных документов.	Sales Invoice Header / Sales Invoice Line, Issued Fin. Charge Memo Header / Issued Fin. Charge Memo Line, Issued Reminder Header / Issued Reminder Line

Характеристики таблицы Master

Таблица Master содержит информацию об основном предмете своей функциональной сферы. Например, таблица **Customer**, которая является основным предметом функциональной области Sales & Receivables, является таблицей Master.

Говоря в целом, много других таблиц связано с таблицей Master, а сама таблица Master связана со многими другими таблицами (обычно таблицами Supplemental).

Обычно имеется таблица Ledger, которая связана с таблицей Master. Кроме того, таблицы Master содержат много динамических фильтров FlowFilters и динамических полей FlowFields, большинство из которых относится к соответствующей таблице Ledger.

Большинство функциональных областей содержит только одну таблицу Master, хотя иногда бывает две таблицы Master для одной функциональной области.

Присвоение имен таблицам Master

Имя этой таблицы – это имя одной из записей таблицы.

Например, таблица **Customer** названа так потому, что каждая содержащаяся в ней запись содержит информацию об одном клиенте.

Первичный ключ (Primary Key) и другие стандартные поля

Первичный ключ таблицы Master – это поле Code длиной 20 символов, с именем «No.» или иногда «Code».

Поле описания этой таблицы – это поле Text длиной 30 символов с именем «Name» или «Description».

Эти два поля входят в свойство DataCaptionFields таблицы, благодаря чему они отображаются в строке заголовка форм этой таблицы.

Ассоциированные формы

Существует три или более форм, используемых совместно с таблицей Master.

- картотечная форма (Card);
- списочная форма (List);
- статистическая форма (Statistics).

Ассоциированная картотечная форма

Прежде всего, существует картотечная форма, которая используется для редактирования записей в таблице Master.

Имя этой формы будет именем таблицы, за которым следует слово «Card». Таким образом, картотечная форма для таблицы **Customer** будет носить имя **Customer Card**.

Она будет содержать по крайней мере одну дополнительную кнопку меню (помимо обычных кнопок OK, Cancel и Help), и у нее будет такое же имя, как у таблицы. Эта кнопка меню обеспечит связь с другими формами, предоставляющими соответствующую или дополнительную информацию. Одна из этих возможностей выбора – связь со списочной формой, которую можно запустить, нажав клавишу F5 на клавиатуре. Другая из этих возможностей выбора – связь с формой, показывающей соответствующую таблицу Ledger, которую можно запустить, нажав сочетание клавиш Ctrl+F5 на клавиатуре. Еще одна из этих возможностей выбора – связь с соответствующей статистической формой, которую можно запустить, нажав клавишу F9 на клавиатуре.

Ассоциированная списочная форма

Вторая форма, используемая совместно с таблицей Master, – это списочная форма, которая используется для просмотра записей в таблице Master. В отличие от картотечной формы, эта форма не может использоваться для редактирования таблицы Master.

Именем этой формы будет имя таблицы, за которым следует слово «List». Таким образом, списочная форма для таблицы **Customer** будет носить имя **Customer List**.

Она будет содержать по крайней мере одну дополнительную кнопку меню (помимо обычных кнопок OK, Cancel и Help), и у нее будет такое же имя, как у таблицы. Она обеспечит связь с другими формами, предоставляющими соответствующую или дополнительную информацию. Одна из этих возможностей выбора – связь с картотечной формой, которую можно запустить, нажав сочетание клавиш Shift+F5 на клавиатуре. Другая из этих возможностей выбора – связь с формой, показывающей соответствующую таблицу Ledger, которую можно запустить, нажав сочетание клавиш Ctrl+F5 на клавиатуре. Еще одна из этих возможностей выбора - связь с соответствующей статистической формой, которую можно запустить, нажав клавишу F9 на клавиатуре.

Эта форма определяется как свойство LookupFormID и свойство DrillDownFormID таблицы, поскольку она используется не только для просмотра, но и для Lookups и Drill downs в этой таблице.

Ассоциированная статистическая форма

Третья форма, используемая совместно с таблицей Master, – это статистическая форма, которая используется для просмотра вычисленных данных о записи в таблице Master. Эта информация передается в отдельную форму из картотечной формы для повышения производительности после того, как эти данные вычисляются.

Именем этой формы будет имя таблицы, за которым следует слово «Statistics». Таким образом, статистическая форма для таблицы **Customer** будет носить имя **Customer Statistics**.

Характеристики таблицы Supplemental

Таблица Supplemental – это таблица, в которой хранится информация о дополнительном предмете, используемом в одной или нескольких функциональных областях.

В качестве примера можно привести таблицу Currency, которая содержит информацию о различных валютах. Это не основной предмет какой-либо функциональной области, но он важен практически для всех них. Говоря в целом, таблицы Supplemental не связаны с другими таблицами, хотя многие другие таблицы всех типов связаны с таблицами Supplemental.

Присвоение имен таблицам Supplemental

Имя этой таблицы – это имя одной из записей таблицы. Например, таблица Currency названа таким образом, потому что каждая содержащаяся в ней запись содержит информацию об одной валюте.

Primary Key и другие стандартные поля

Первичный ключ таблицы Supplemental – это поле Code длиной 10 символов с именем «Code».

Поле описания этой таблицы – это поле Text длиной 30 или 50 символов с именем «Description». Некоторые таблицы Supplemental не содержат поле описания, но некоторые содержат поле описания с именем «Name».

Ассоциированные формы

Как описано выше, форма, используемая для таблицы Supplemental - это табличная форма (Tabular). Имя этой формы – это имя таблицы Supplemental во множественном числе. Поэтому форма, которая используется для редактирования таблицы Currency, называется «Currencies».

Эта форма обычно не содержит кнопок, не считая стандартных кнопок OK, Cancel и Help. Данная форма устанавливается как свойство LookupFormID таблицы, так как она используется не только для редактирования, но и для поиска в этой таблице.

Характеристики таблицы Subsidiary

Таблица Subsidiary содержит информацию, являющуюся вспомогательной либо для таблицы Master, либо для таблицы Supplemental, либо для них обеих.

Например, таблица **Item Vendor** – это таблица, содержащая вспомогательную информацию (номера поставщиков) для таблицы **Item** в функциональной области Inventory.

Primary Key и другие стандартные поля

Первичный ключ для таблицы Subsidiary будет содержать поле для каждой таблицы, по отношению к которой он является вспомогательным, и каждый из них связан с этой таблицей.

Например, поля Primary Key для таблицы **Item Vendor** содержат поле **Item No.** (связанное с **Item т**аблицы Master) и поле **Vendor No.** (связанное с **Vendor** таблицы Master).

Первичный ключ может также содержать Integer как последнее поле (с именем **Line No.**), что позволит дифференцировать несколько записей с таким же вспомогательным отношением. Например, таблица **Employee Qualification** содержит поле **Integer** в первичном ключе для дифференциации двух записей для одного и того же сотрудника (Employee).

Вспомогательные таблицы обычно не относятся к другим таблицам, за исключением полей первичного ключа, упомянутых выше. Другие таблицы обычно не относятся к таблице Subsidiary вследствие того, что каждая таблица Subsidiary содержит несколько полей в первичном ключе.

Таблицы Subsidiary обычно не содержат полей с описанием.

Присвоение имен таблицам Subsidiary

Имя этой таблицы обычно составляется из имен таблицы или таблиц, по отношению к которым она является вспомогательной или наподобие этого.

Например, таблица, которая является вспомогательной по отношению и к таблице **Vendor,** и к таблице **Item**, называется **Item Vendor**. Имя всегда стоит в единственном числе и описывает одну запись, содержащуюся в таблице.

Ассоциированные формы

В таблице Subsidiary используется одна форма как для редактирования, так и для просмотра, и эта форма обычно вызывается с помощью кнопки в форме Master или Supplemental, по отношению к которой она является вспомогательной.

Обычно имя этой формы – это имя таблицы во множественном числе.

Если первичный ключ для таблицы Subsidiary содержит Integer, эта форма будет рабочей формой (Worksheet), и она не будет содержать ни одного поля первичного ключа. Вместо этого, поля первичного ключа, кроме Integer, будут отфильтрованы так, что они будут просматриваться и устанавливаться автоматически.

Если первичный ключ для таблицы Subsidiary не содержит Integer, эта форма будет табличной формой, и она не будет содержать поле первичный ключ основной таблицы. Вместо этого данное поле будет отфильтровано так, чтобы оно просматривалось и устанавливалось автоматически.

В обоих случаях форма связана с формой, вызвавшей ее, поэтому, если запись в ней была изменена, фильтры этой формы тоже изменяются. Это гарантирует, что всегда будут доступны корректные связанные записи.

Характеристики таблицы Ledger

Таблица Ledger содержит информацию об операциях, являющуюся основным предметом ее функциональной области Например, таблица **Customer Ledger Entry**, содержащая всю информацию о сделках Sales и Receivables и являющаяся основным предметом функциональной области Sales и Receivables, является таблицей Ledger.

Эта таблица очень похожа на таблицу Subsidiary, так как связана с соответствующей таблицей Master, но у нее другие характеристики, как мы увидим позже. Она также связано со многими другими таблицами, в основном с таблицами Supplemental. Таблицы Register связаны с таблицами Ledger, но другая таблица редко бывает связана с таблицей Ledger.

Обычно существует одна таблица Ledger на одну функциональную область, хотя иногда бывает и две.

Присвоение имен таблицам Ledger

Имя таблицы Ledger – это обычно имя таблицы Master, с которой она связана, плюс слова «Ledger Entry», описывающие одну из имеющихся в ней записей (запись). Длинное имя может быть сокращено.

Например, таблица **Customer Ledger Entry** на самом деле называется **Cust. Ledger Entry**.

Иногда, при наличии нескольких таблиц Master, имя – это имя функциональной области, за которым следуют слова «Ledger Entry». Таким образом, таблица Ledger для функциональной области Payroll называется **Payroll Ledger Entry**.

Таблицы Ledger не могут модифицироваться пользователями, если не считать некоторых тщательно контролируемых исключений.

В число этих исключений никогда не входит возможность добавления или удаления записи.

Primary Key и другие стандартные поля

Первичный ключ таблицы Ledger – это Integer с именем «Entry No.». Этот первичный ключ всегда генерируется автоматически стандартной программой регистрации, управляющей данной таблицей Ledger.

В таблице Ledger всегда есть поле, которое имеет отношение таблицы к таблице Master, связанной с этой таблицей Ledger.

Поле описания этой таблицы – это поле Text длиной 50 символов с именем «**Description**».

В дополнение к первичному ключу таблицы Ledger обычно содержат много вторичных ключей, ко многим из которых присоединены поля SumIndex. Они используются совместно с динамическими полями FlowFields в таблице Master для вычисления данных для пользователя. Вследствие этого, по крайней мере один, а обычно несколько вторичных ключей содержат поле, которое связано с таблицей Master как первое поле в ключе.

Ассоциированные формы

В таблице Ledger используется списочная форма для отображения информации для пользователя. Эта форма имеет имя таблицы

Ledger во множественном числе. Например, имя формы, которая отображает записи из таблицы **Customer Ledger Entry** – это **Customer Ledger Entries**.

Эта форма определяется как свойство LookupFormID и свойство DrillDownFormID таблицы, поскольку она используется не только для просмотра, но и для Lookups и Drill downs в этой таблице.

Эта форма может отображаться из форм таблицы Master при нажатии сочетания клавиш Ctrl+F5 на клавиатуре.

Характеристики таблицы Register

Таблица Register – это «содержание» для соответствующей таблицы или таблиц Ledger.

Существует одна запись в процессе проводки, и таблица Register больше соответствует стандартной программе регистрации, чем функциональная область.

Например, таблица, содержащая список записей, внесенных в таблицу **Customer Ledger Entry,** - это таблица **G/L Register** . Это – следствие того, что Customer Ledger Entries зарегистрированы из таблицы General Journal с использованием процедур регистрации General Journal. Register Table связана с соответствующей таблицей или таблицами Ledger .

Присвоение имен таблицам Register

Таблицам Register имена присваиваются согласно функции регистрации, за которой следует слово «Register». Таким образом, таблице Register, обновленной функцией регистрации General Ledger, присваивается имя **G/L Register.**

Пользователь не может изменять таблицу Register.

Primary Key и другие стандартные поля

Первичный ключ таблицы Ledger – это Integer с именем «No.». Этот первичный ключ всегда генерируется автоматически стандартной программой регистрации, управляющей этой таблицей Register.

Другие стандартные поля для таблицы Register содержат два поля, оба поля типа Integer, связанных с соответствующей таблицей Ledger, с именем «From Entry No.» и «To Entry No.».

Таблицы Register обычно не содержат полей с описанием.

Ассоциированные формы

В таблице Register используется списочная форма для отображения информации для пользователя.

Эта форма обычно имеет такое же имя, как таблица Register.

Она содержит кнопку меню с именем «Register», которая содержит ссылки на другие списочные формы, отображающие соответствующую таблицу или таблицы Ledger.

Характеристики таблицы Journal

Таблица Journal – это таблица для ввода основной информации о торговых операциях для функциональной области.

Все операции, как введенные самим пользователем, так и сгенерированные другой стандартной программой регистрации, проходят через эту таблицу, чтобы в конечном итоге появиться в таблице Ledger.

Таблицы Journal связаны со многими другими таблицами, включая таблицу Master, таблицы Supplemental, таблицы Subsidiary и даже соответствующую таблицу Ledger (для прикладных целей). Другие таблицы не связаны с таблицей Journal.

Поскольку таблицы Journal используются для записей операций, они содержат большее количество триггерного кода, чем большинство других типов таблиц.

Присвоение имен таблицам Journal

Имя этой таблицы – это имя типа регистрируемой операции, за которым следуют слова «Journal Line».

Например, имя таблицы, в которую пользователь вводит операции в функциональную область Resource – это **Resource Journal Line**. Каждая запись таблицы journal содержит одну строку из journal.

Таблица Journal обычно относится к двум соответствующим таблицам Supplemental – таблице **Journal Template** и таблице **Journal Batch**. Эти таблицы позволяют пользователям разбивать записи данных

различными способами и позволяют им устанавливать дополнительную информацию, имеющую отношение ко всей таблице Journal.

Имена этих двух таблиц такие же, как имя таблицы Journal, за исключением того, что за ними следуют слова Journal Batch или Journal Template, а не Journal Line. Поэтому эти две соответствующих Resource Journal Line таблицы получили имена **Resource Journal Template** и **Resource Journal Batch**.

Primary Key и другие стандартные поля

Первичный ключ для таблицы Journal составлен из трех полей – поля, связанного с таблицей Journal Template, поля, связанного с таблицей Journal Batch, и поля типа Integer с именем **Line No**.

Поле с описанием этой таблицы – это поле Text длиной 50 символов с именем «**Description**».

Ассоциированные формы

Форма, которая используется для таблицы Journal – это рабочая форма, которая используется для ввода записей в эту таблицу.

Эта форма имеет то же имя, что и таблица, но в нем нет слова «Line». Поэтому рабочая форма для таблицы **Resource Journal Line** называется **Resource Journal**. Иногда форма получает имя по типу вводимых данных. Например, одна из многих рабочих форм, используемых совместно с таблицей **Gen. Journal Line,** называется **Sales Journal**.

Ни одно из полей первичного ключа не включено в эту форму.

При вызове оно фильтруется полями Template и Batch, и свойство AutoSplitKey рабочей формы автоматически устанавливает поле **Line No**.

Рабочая форма всегда содержит кнопку меню, которая имеет то же имя, что и связанная таблица Master. Она включает возможность вызова картотечной формы для таблицы Master, которую можно запустить с помощью сочетания клавиш Shift+F5 на клавиатуре. Кроме того, она включает возможность вызова таблицы Ledger для той текущей записи Master, которую можно запустить с помощью сочетания клавиш Ctrl+F5 на клавиатуре.

Кроме того, рабочая форма всегда содержит кнопку меню, которая имеет имя «Posting» и позволяет пользователю распечатать отчет о тесте, зарегистрировать или зарегистрировать и распечатать реестр, и иногда включает возможность восстановления. Рабочая форма также обычно содержит другие кнопки, что позволяет пользователю выполнять различные функции обработки.

Характеристики таблицы Document

Таблицы Document – это таблицы вторичных операций, которые допускают ввод в функциональную сферу или сразу в несколько функциональных сфер Они вторичны только в том смысле, что их данные регистрируются в ledgers через таблицы journal, а не непосредственно.

Для большинства пользователей это основное средство ввода данных, поскольку его используют чаще всего. Поскольку таблицы Document используются для записи операций, они содержат большее количество триггерного кода, чем большинство других типов таблиц.

Фактически существует два типа таблиц Document:

- таблица Document Header;
- таблица Document Line.

Таблица Document Header

Таблица Document Header содержит основную информацию об операциях. Например, при совершении коммерческой сделки таблица **Sales Header** содержит информацию, имеющую отношение к заказу или счету-фактуре в целом.

Таблица Document Line

Таблица Document Line содержит подробную информацию о транзакции. Например, в случае проведения торговой операции таблица **Sales Line** содержит информацию о каждой отдельной строке заказа или счета-фактуры. Таблица Document Line – это таблица Subsidiary по отношению к таблице Document Header.

Подобно таблицам Journal, таблицы Document связаны со многими другими таблицами, в том числе Master, Supplemental и Subsidiary, но другая таблица редко бывает связана с таблицей Document.

Присвоение имен таблицам Document

Имя таблицы Document Header – это имя операции или документа плюс слово «Header». Например, таблица Document Header, которая содержит операции по продажам, называется **Sales Header**. Каждая запись содержит одну продажу (заказ или счет-фактуру).

Еще один пример. Таблица Document Header, содержащая операции Finance Charge Memo, называется **Finance Charge Memo Header**. Каждая запись содержит одну Finance Charge Memo.

Имя таблицы Document Line – это имя операции или документа плюс слово «Line». Например, таблица Document Line, содержащая операции по продажам, называется **Sales Line**. Каждая запись содержит одну строку из продажи (заказ или счет-фактуру).

Другой пример. Таблица Document Line, которая содержит операции Finance Charge Memo, называется **Finance Charge Memo Line**. Каждая запись содержит одну строку из Finance Charge Memo.

Первичный ключ и другие стандартные поля

Для большинства таблиц Document Header, первичный ключ – это поле Code длиной 20 символов, содержащее номер документа и имя **No**.

Некоторые таблицы Document Header содержат много типов документов, каждый из которых относится к одной и той же операции. Например, таблица Sales Header содержит документы Invoice, Credit Memo, Sales Order и так далее. В таких случаях первичный ключ содержит два поля – поле Option с именем Document Type и поле Code длиной 20 символов с именем **No**.

Для большинства таблиц Document Line, первичный ключ содержит два поля – поле Code длиной 20 символов, содержащее номер документа, и поле Integer с именем **No**. Поле Code – это таблица, связанная с таблицей Document Header и названная согласно имени таблицы (без слова «Header»), за которым следует поле первичный ключ этой таблицы. Например, поле Code в первичном ключе таблицы Finance Charge Memo Line получило имя **Finance Charge Memo No**.

В случае, когда Document Header имеет тип Document, первичный ключ таблицы Document Line содержит 3 поля – поле Option с именем **Document Type**, поле Code длиной 20 символов с именем **Document**

No. и поле Integer с именем **Line No**. Поле Code – это таблица, связанная с таблицей Document Header.

Ассоциированные формы

В таблице Document Header используется специальный вид картотечной формы для отображения одной записи заголовка единовременно, чтобы пользователь мог просматривать и редактировать данные.

Имя этой формы – это имя документа, который она отображает. Так происходит, даже если таблица может содержать много типов документов, поскольку, в этом случае, форма настроена только на просмотр информацию из одного типа. Она содержит вкладки для разбиения полей на логические группы, чтобы облегчить пользователю задачу редактирования данных.

Форма Document Header всегда содержит кнопку меню, которая называется «Posting», позволяющую пользователю печатать отчеты о тестах, регистрировать, регистрировать и печатать Document или регистрировать пакет этих документов.

Кроме того, эта форма обычно содержит другие кнопки, что позволяет пользователю выполнять различные функции обработки.

Форму Document Header отличает от картотечной формы то, что она содержит сопутствующую форму, которая, в свою очередь, содержит форму Document Lines.

В таблице Document Line используется специальный вид рабочей формы, предназначенный для отображения нескольких строк записей за один раз, чтобы пользователь мог их просматривать и редактировать по одной.

Имя формы – это имя документа, за которым следует либо слово «Lines», либо «Sub-Form». Поскольку имя никогда не отображается, это не столь критично.

Действительно критично то, что размер этой формы является точно таким же, как размер сопутствующей формы, в которой она отображена.

Ни одно из полей первичного ключа не должно быть включено в эту форму, поскольку форма фильтруется (с помощью ссылки) для всех

полей первичного ключа, кроме поля Integer, которое обрабатывается автоматически свойством AutoSplitKey рабочей формы.

В таблице Document Header также используется списочная форма, что позволяет пользователю просматривать несколько документов одновременно.

Имя этой формы, как и у большинства списочных форм, – это имя таблицы (без слова «Header») за которым следует слово «List». Эта списочная форма может вызываться из формы Header нажатием клавиши F5 на клавиатуре.

Характеристики таблиц Document Table

Таблицы Document History являются тем же самым для таблиц Document, что таблицы Ledger – для таблиц Journal.

Когда Document регистрируется, частью этого процесса регистрации становится копирование таблиц Document в соответствующие таблицы Document History.

Для облегчения процесса копирования таблица Document History обычно имеет те же поля с теми же номерами полей, именами и свойствами, что и оригиналы таблиц Document.

Поскольку таблицы Document History записывают зарегистрированные операции, они обычно не доступны для редактирования пользователем, хотя могут быть удалены, если пользователь имеет на то разрешение.

Таблицы и формы Document History, отличающиеся от упомянутых выше, – такие же, как соответствующие таблицы Document.

Действительно, сопутствующая форма – это скорее списочная форма, чем рабочая форма, поскольку изменения здесь не разрешены.

И имена содержат либо слово «Posted», либо «Issued» для указания того, что это – хронологическая таблица, а не таблица для ввода операций.

Характеристики таблицы Setup

Таблица Setup – это таблица, которая предназначена для хранения только одной записи. Этот запись содержит различные поля, которые

используются для выбора параметров системы Navision или для хранения данных, которые относятся к компании в целом.

Обычно к таблице Setup не относятся никакие таблицы, хотя таблицы Setup могут быть связаны с другими таблицами, обычно таблицами Supplemental.

Присвоение имен таблицам Setup

Обычно имя этой таблицы – это имя функциональной области, которую она должна устанавливать, за которым следует слово «Setup». Например, таблица, содержащая информацию для установки функциональной области General Ledger, называется **General Ledger Setup**. Одно исключение из этого правила - таблица **Company Information**.

Primary Key и другие стандартные поля

Первичный ключ этой таблицы –поле Code длиной 10 символов с именем **Primary Key**. Он всегда остается пустым, так как разрешена только одна запись для каждой таблицы. У этого типа таблицы нет поля с описанием.

Ассоциированная установочная форма

Для этой таблицы установлена только одна форма, это – установочная форма (Setup), как описано в следующем разделе. Эта форма имеет такое же имя, как таблица. Поле первичного ключа не включено в эту форму.

Примечание

Существуют таблицы, имеющие слово «Setup» в своем имени и содержащие несколько записей. Эти таблицы обычно подчиняются правилам таблиц Subsidiary, описанных ранее, а не правилам, раскрытым здесь.

2.9 Ответы на вопросы

На этом раздел упражнений с таблицами завершается. Если вы не все поняли из того, что делали в этой главе, перечитайте разделы о таблицах и затем снова выполните упражнения.

Крайне важно, чтобы вы усвоили концепции, о которых узнали в этой главе. Таблицы – это основа, на которой построено все остальное в системе Navision.

Здесь даны ответы на все вопросы, поставленные в предыдущих разделах.

Упражнение 2.2.1 Изучение существующих таблиц

Какова опциональная строка поля **Reserve** в таблице **Customer**?

Never,Optional,Always

Каково различие между формулами для вычислений поля **Balance** и поля **Balance At Date** в таблице **G/L Account** ?

Поле Balance At Date содержит еще одну строку фильтра в Table Filter, куда входит Date Filter.

Какое поле является полем **Date Filter** в таблице **G/L Account** ?

Date, FlowFilter

Каково значение свойства TableRelation поля **Currency Code** в таблице **Customer**?

Currency.

Упражнение 2.3.1 Создание простой таблицы

Здесь нет вопросов.

Упражнение 2.3.2 Добавление данных

С какой проблемой вы столкнулись?

Невозможно ввести вторую запись. Вы получаете сообщение об ошибке. Запись уже существует.

Почему вы не можете ввести вторую запись?

Primary key – это только поле Model. Это первое поле в таблице, следовательно, поле Model должно быть уникальным для каждой записи.

Упражнение 2.4.1 Настройка первичного ключа

Здесь нет вопросов.

Упражнение 2.4.2 Создание вторичных ключей

Здесь нет вопросов.

Упражнение 2.4.3 Использование вторичных ключей

Посмотрите на записи, которые содержат дату производства (Date of Mfg.), равную 01.15.00. В каком порядке расположены эти записи, если они содержат одну и ту же дату?

Они отсортированы по Transmission, второму полю во вторичном ключе.

Посмотрите на записи, содержащие одни и те же Date of Mfg. и Transmission. В каком порядке расположены эти записи?

Они отсортированы по первичному ключу.

Что это говорит вам о вторичных ключах?

Если поле Primary Key не используется во вторичном ключе, оно добавляется в конец вторичного ключа.

Упражнение 2.5.1 Простая таблица с отношениями таблиц

Введите «a» в поле **Salesperson Code**. Что произойдет при нажатии клавиши Enter?

Клиент находит первое значение в таблице, содержащее букву “a” в начале, и выбирает для вас эту запись.

Введите «XY» в поле **Salesperson Code**. Что произойдет при нажатии клавиши Enter?

Вы получите ошибку, потому что XY не существует в таблице Salesperson/Purchaser.

Упражнение 2.5.2 Переименование записей

Сколько времени потребовалось, чтобы изменить это значение? Больше, чем вы ожидали?

На это ушло несколько секунд.

Да.

Проверьте данные в таблице 91300 (**Sales Transaction**), открыв ее. Изменились ли какие-либо записи здесь? Почему?

Да. Все записи, содержавшие мои инициалы, изменились на XXX.

Потому что переименование записи приводит к изменению всех значений, относящихся к ней, в других таблицах.

Упражнение 2.5.3 Фильтрованные отношения таблиц

Когда вы используете кнопку поиска для поиска в поле **Salesperson Code**, все ли записи в таблице **Salesperson/Purchaser** вы видите? Если нет, почему?

Нет. Фильтр таблицы по отношению таблиц отфильтровал все записи, где Commission % <= 0.

Можете ли вы ввести значение *AH* в поле **Salesperson Code**? Если нет, то почему?

Нет. Отношение таблиц с фильтром больше не признает AH в качестве допустимого значения, потому что оно не существует в наборе записей с фильтром.

Упражнение 2.5.4 Условные отношения таблиц

Когда вы используете кнопку поиска в поле **No.**, какая форма отображается? Что произойдет, если вы измените поле **Type**?

Отношение таблиц теперь зависит от поля Type. Использование кнопки Lookup при Type=Item приводит к отображению формы Item List и т.д.

Замените поле Type на Item и попробуйте найти номер товара. Теперь замените поле Type на *Resource*. Что произошло? Является ли допустимым значение для поля No.?

Ничего не происходит!

Нет. Значение в No. – это Item No., а не Resource No. Это недопустимо.

Теперь заново введите значение в поле **No.** после замены Type, как описано в предыдущем вопросе. Какую ошибку вы получаете?

Resource No. «70000» не существует.

Упражнение 2.6.1 Вычисление сумм вручную

Какая сумма	**Общее количество**
Для JR	1050.00
Для MF	800.00
Для PS	185.00
Для BCEX	2035.00

Упражнение 2.6.2 Создание поля FlowField

Здесь нет вопросов.

Упражнение 2.6.3 Тестирование поля FlowField

Что произошло, когда вы попытались открыть таблицу?

Получена ошибка.

Чем было вызвано сообщение об ошибке?

Отсутствующий Key и SumIndexField

Что мы должны изменить, чтобы решить проблему (подсказка: очень внимательно прочитайте сообщение)?

Необходимо добавить ключ в таблицу Sales Transaction, содержащий корректные поля и SumIndexField of Amount.

Упражнение 2.6.4 Создание поля SumIndexField

Были ли какие-либо ошибки на этой раз? Почему были или почему не были?

Нет.

Обнаружены ключ и SumIndexField, необходимые для вычисления FlowFields.

Какая сумма	Продажи
Для JR	2035.00
Для MF	2035.00
Для PS	2035.00

Верна ли общая сумма? Какие есть подозрения на их счет?

Нет.

Они все совершенно одинаковы.

Соответствуют ли они какому-либо из значений из упражнения 5.4.1 (суммы, вычисленные вами вручную)? Каким значениям они соответствуют?

Да.

Они соответствуют общему количеству для всей таблицы.

Что, как вам кажется, неверно в вашем поле FlowField?

Оно не суммирует записи для каждого продавца. Оно суммирует всех продавцов.

Упражнение 2.6.5 Фильтрация поля FlowField

Что произошло, когда вы попытались открыть таблицу?

Получена ошибка.

Чем было вызвано сообщение об ошибке?

Отсутствующие Key и SumIndexField

Что мы должны изменить, чтобы решить проблему (подсказка: очень внимательно прочитайте сообщение)?

Необходимо добавить ключ в таблицу Sales Transaction, содержащий корректные поля и SumIndexField of Amount.

Упражнение 2.6.6 Процесс создания поля SumIndexField

Имя таблицы: Sales Transaction

Имя поля: Amount

Имя поля 1: Salesperson Code

Были ли какие-либо ошибки на этой раз? Почему были или почему не были?

Нет.

Поля SumIndexField and FlowField соответствовали.

Упражнение 2.6.7 Создание FlowFilter

Здесь нет вопросов.

Упражнение 2.6.8 Использование FlowFilter

Что произошло?

Произошла ошибка.

Почему вы получили сообщение только после того, как установили FlowFilter?

Специальный ключ не требуется до тех пор, пока не будет использовано поле FlowFilter.

Какое сообщение об ошибке вы получили?

Не удается вычислить FlowFilter. Вы должны установить и активировать ключ …

Вы видели раньше подобное сообщение?

Да. Перед тем, как было установлено поле SumIndexField.

Какое действие вы должны предпринять дальше, чтобы решить проблему?

Добавить поле Add the Type для моего поля SumIndexField.

Упражнение 2.6.9 Решение проблемы.

Все ли работало правильно тогда?

Да. Поле FlowFields отфильтровано нужным образом.

Упражнение 2.7.1 Свойство NotBlank

Какое «имя» имеет эта запись (помните, что имя – это значение первичных полей ключа)?

Model = '', Serial No. = 0

Что произойдет, если вы повторите действия 2-5, описанные выше?

Выдается сообщение об ошибке с информацией о том, что запись уже существует.

Как мы могли бы удалить пустые записи, помещаемые в таблицу случайно?

Принудить пользователя заполнять хотя бы поля первичного ключа.

Что происходит на этот раз?

Мне приходится ввести Model. Я не могу оставить его пустым.

Что вы должны сделать теперь, чтобы вставить новую запись?

Мне нужно ввести значение хотя бы для поля Model.

Упражнение 2.7.2 Свойство Editable

Что произошло?

Значение изменилось на 500. Произошла ошибка.

Сохранилось ли изменение (если вы закрыли таблицу и снова открыли ее, значение PS по-прежнему 500)?

Да.

Какова новая сумма? Какой был старая сумма (раздел 5.4.1)? В чем различие?

Новая: 500

Старая: 185

Разница: 315

Можете ли вы определить, какая строка привела к изменению суммы?

Да. Там есть новая строка с количеством 315 для PS.

Вы ввели эту строку? Как она была введена?

Нет.

Очевидно, эта запись была вставлена при изменении поля FlowField.

Упражнение 2.7.3 Свойство DecimalPlaces

Удалось ли вам правильно ввести количество?

Произошла ошибка.

Какую ошибку вы получили?

Поле может содержать максимум 2 десятичных разряда.

Что произошло?

Мне удалось ввести значение со всеми тремя десятичными разрядами.

Что отображается в поле, если вы вводите количество *234,1* и нажимаете клавишу Enter?

234.10

Что отображается в поле, если вы вводите количество *234* и нажимаете клавишу Enter?

234.00

Что отображается в поле, если вы вводите количество *234,110* и нажимаете клавишу Enter?

234.11

2.10 Контрольные вопросы

В этом разделе делается обзор концепций, освещаемых в главе, посредством постановки вопросов. Ответы даны в подразделе «Ответы».

Вопросы

1 Какие два основных пункта вы можете определить в описании таблицы?

2 Какое свойство таблицы определяет форму поиска по умолчанию для таблицы?

3 Какое свойство поля используется, чтобы принудить пользователя ввести значение в поле первичного ключа?

4 Какой диапазон чисел вы можете использовать для новых полей, которые вы добавляете в таблицу **Customer**?

5 Если вы хотите просматривать клиентов, сортируя их по городу, в котором они живут, что вам надо изменить в описании таблицы?

6 Как бы вы изменили поле **Salesperson Code** в таблице **Customer**, чтобы пользователь мог подбирать только тех продавцов, у которых ставка Commission % больше нуля?

7 Каким типом таблицы является таблица **Customer**?

8 Каким типом таблицы является таблица **Item Vendor**?

9 Каким типом таблицы является таблица **Currency**?

10 Каким типом таблицы является таблица **Item Ledger Entry**?

Ответы

1 Какие два основных пункта вы можете определить в описании таблицы?

Поля и Ключи.

2 Какое свойство таблицы определяет форму поиска по умолчанию для таблицы?

LookupFormID

3 Какое свойство поля используется, чтобы принудить пользователя ввести значение в поле первичного ключа?

NotBlank устанавливается на Yes

4 Какой диапазон чисел вы можете использовать для новых полей, которые вы добавляете в таблицу **Customer**?

От 50000 до 99999

5 Если вы хотите просматривать клиентов, сортируя их по городу, в котором они живут, что вам надо изменить в описании таблицы?

Добавить вторичный ключ, содержащий поле City в качестве первого поля.

6 Как бы вы изменили поле **Salesperson Code** в таблице **Customer**, чтобы пользователь мог подбирать только тех продавцов, у которых ставка Commission % больше нуля?

Следует добавить фильтр таблицы в свойство отношения таблиц, которое фильтрует продавцов, где Commission % > 0.

7 Каким типом таблицы является таблица **Customer**?

Таблица Master функциональной области Sales и Receivables.

8 Каким типом таблицы является таблица **Item Vendor**?

Вспомогательная по отношению к таблицам Vendor и Item.

9 Каким типом таблицы является таблица **Currency**?

Дополнительная по отношению ко всем функциональным областям.

10 Каким типом таблицы является таблица **Item Ledger Entry**?

Ledger по отношению к функциональной области Inventory

Глава 3

Основные формы

Формы используются для ввода и отображения данных. Например, вы можете 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. Упражнения, содержащиеся в данной главе, помогут укрепить знания, которое вы получите в этой части курса.

Глава содержит следующие разделы.

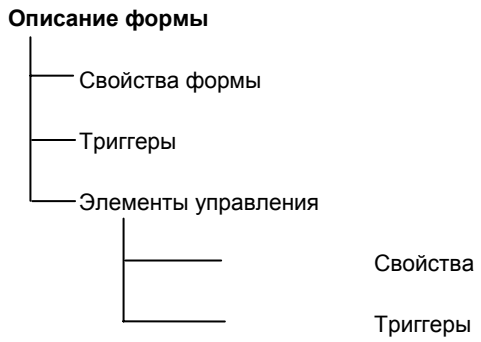
- Что такое формы;
- Работа с пустой формой;
- Другие полезные элементы управления;
- Использование мастера форм;
- Создание кнопок;
- Создание дополнительных форм;
- Типы и характеристики форм;
- Стандартные средства навигации;
- Ответы на вопросы;
- Обзор главы.

3.1 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, и 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. В этой и последующих главах проводится детальное исследование каждого компонента.



Формы создаются и редактируются в Дизайнере форм (Form Designer).

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. Фрейм сам по себе не отображает данные или информацию, но он может содержать ряд других элементов управления, которые вы хотите сгруппировать. Мощным контейнерным элементом управления является закладка (tab control). Закладка – это по сути ряд фреймов или страниц, которые помещены друг над другом. Пользователь может переключаться между страницами, щелкая снабженные заголовками закладки. 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, которые связаны с таблицей, обычно связаны с полями в этой же таблице. 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; 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.

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 – тип элемента 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 появится возможность вставлять новые и обновлять существующие записи.

Свойства определяются в окне **Properties**, которое можно редактировать, когда форма открыта в Дизайнере форм.

Что такое триггеры

Некоторые предопределенные события (events), которые происходят с формой или элементом управления, позволяют системе выполнять определяемую пользователем C/AL-функцию – событие включает функцию. Событие и функция в совокупности называются триггером (trigger). Триггеры форм содержат функцию OnOpenForm, имеющую инструкции, которые будут выполняться при открытии формы, и функцию OnModifyRecord, имеющую инструкции, которые будут выполняться до того, как система примет вводимые пользователем в запись изменения. Триггеры редактируются в Редакторе C/AL-кода, который можно открыть из Дизайнера форм.

3.2 РАБОТА С ПУСТОЙ ФОРМОЙ

В этом разделе вы научитесь создавать пустую форму. Это не лучший способ создания формы, но он позволит вам глубже разобраться с работой мастера форм (Form Wizard) и Field Menu (они описаны в последующем разделе).

Упражнение 3.2.1 Создание пустой формы

Вы начнете с создания пустой формы. Есть мастер, который поможет вам создавать формы, но вы будете пользоваться им позже. Последовательность действий при создании пустой формы такова.

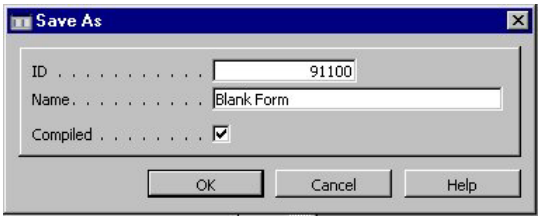
1 Нажмите вариант меню Tools, Object Designer.

2 В окне **Object Designer** нажмите Form.

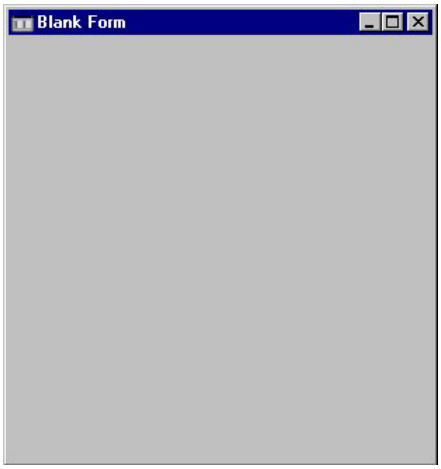
3 Нажмите New. Приложение C/SIDE отобразит окно **Form Wizard**.

4 Нажмите кнопку OK, чтобы создать пустую форму. C/SIDE отобразит окно **Form Designer**. Когда вы создаете пустую форму, ее размер довольно мал. Вы можете увеличить форму, если это необходимо для выполнения упражнений.

5 Закройте, сохраните и откомпилируйте новую форму. Используйте имя «Blank Form» и идентификатор 91100. Вы будете обращаться к этой форме в последующих упражнениях.



6 Откройте форму.



Каково значение в строке заголовка формы (обычно это синяя область в верхней части формы, содержащая кнопки Min, Max и Close)?

Что происходит, когда вы щелкаете кнопку Next на панели инструментов?

Форма считывает данные из таблицы?

Упражнение 3.2.2 Присоединение исходной таблицы

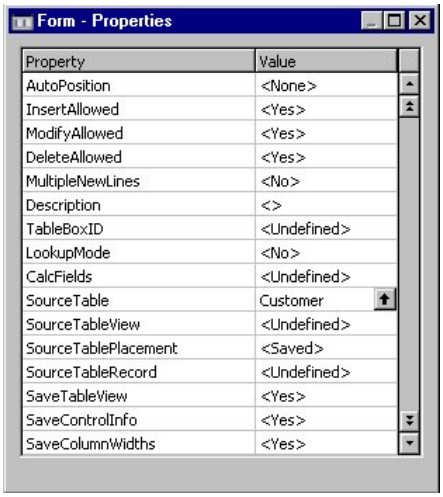
Пустая форма сразу после того, как она создана, еще не является связанной. Это означает, что она не ассоциирована с какой-либо таблицей. В этом упражнении вы свяжете созданную пустую форму с таблицей и посмотрите, что в результате произойдет.

1 Откройте форму 91100 (Blank Form) в Дизайнере форм.

2 Откройте окно **Properties** этой формы.

3 Щелкните свойство SourceTable.

4 Введите число *18* в это свойство. 18 – это идентификатор таблицы **Customer**. Ваша форма теперь связана с таблицей **Customer**.



5 Закройте, сохраните и откомпилируйте форму.

Теперь давайте посмотрим, что изменилось.

Откройте форму 91100 (Blank Form) и ответьте на следующие вопросы.

Каково значение в строке заголовка формы (обычно это синяя область в верхней части формы, содержащая кнопки Min, Max и Close)?

Что происходит, когда вы щелкаете кнопку Next на панели инструментов?

Форма считывает данные из исходной таблицы?

Упражнение 3.2.3 Добавление простых элементов управления

В этом упражнении вы добавите текстовую зону, надпись и выключатель (checkbox) на свою пустую форму. Это наиболее часто используемые элементы управления, содержащиеся на форме.

Добавление текстовой зоны

1 Откройте пустую форму 91100 (Blank Form) в Дизайнере форм.

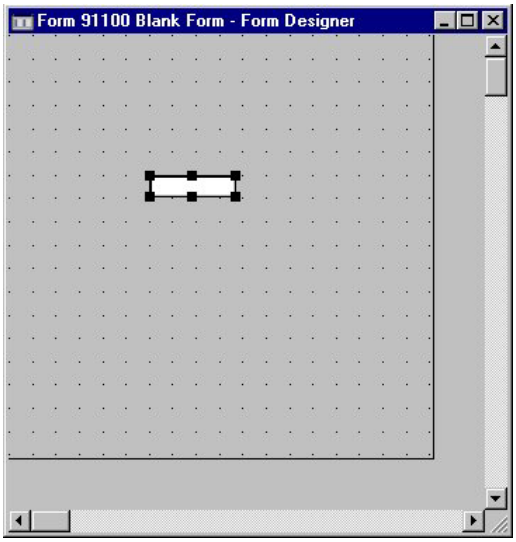
2 Нажмите кнопку Toolbox на панели инструментов (или View, Toolbox). Появится окно **Toolbox**.

3 В течение нескольких секунд подержите курсор мыши над каждой кнопкой в окне **Toolbox**. Благодаря этому вы увидите Tool Tip (поясняющую надпись) для каждой кнопки. Tool Tip отображает имя элемента управления, представляемого кнопкой.

4 Нажмите кнопку Text Box в окне **Toolbox**. Это позволяет выбрать этот тип элемента управления (кнопка остается нажатой). Чтобы отменить выбор этого типа элемента управления, вы можете нажать кнопку Arrow в верхней левой части окна **Toolbox**.

5 При выбранной кнопке Text Box поместите курсор (указатель мыши) над пустой формой. Курсор должен превратиться в перекрестие с прямоугольником.

6 Щелкните где-нибудь в середине формы. Обратите внимание, что там, где вы нажали, будет находиться верхний левый угол текстовой зоны. Если вы переместите мышь, удерживая нажатой кнопку мыши, размер текстовой зоны изменится. Обычно этого делать не нужно.



Теперь форма содержит текстовую зону. Какая информация будет в ней отображаться?

Попытайтесь открыть форму, щелкнув вариант меню File, Run. Что происходит?

Связывание текстовой зоны

Элемент управления “текстовая зона” должен быть связан с полем или другим элементом данных (это может быть то, что возвращает допустимое значение – переменная, функция или какое-то выражение).

Чтобы связать текстовую зону, вы должны установить соответствующее свойство.

1 Нажмите текстовую зону, чтобы ее выбрать.

2 Откройте окно **Properties**.

3 Измените свойство SourceExpr на *«”No”»*. (включая двойные кавычки).

4 Закройте, сохраните и откомпилируйте форму.

5 Откройте форму и проверьте новую текстовую зону, перемещаясь в следующую, предыдущую, первую или последнюю записи.

Добавление надписи

Есть только одна проблема, с которой может столкнуться тот, кто использует вашу очень простую форму. Пользователь может не знать, что в ней отображается. Чтобы ему помочь, вы добавите надпись, которая описывает назначение поля формы.

1 В окне **Toolbox** нажмите Label (не забудьте, что найти нужную кнопку вам поможет Tool Tips).

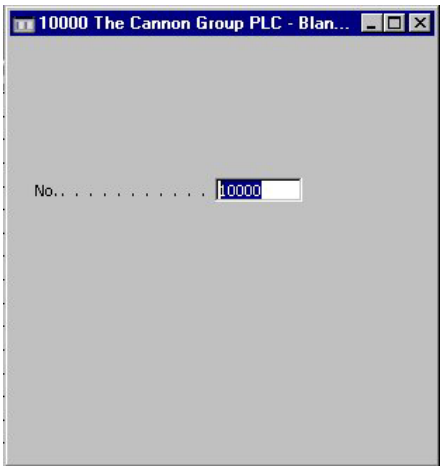
2 Поместите курсор над формой и щелкните где-нибудь слева от текстовой зоны, которую вы добавили.

3 Откройте окно **Properties** этой зоны.

4 Измените свойство Caption на *No*.

5 Закройте, сохраните и откомпилируйте форму.

6 Откройте форму и проверьте новую текстовую зону, перемещаясь в следующую, предыдущую, первую или последнюю запись. Надпись позволяет узнать, какое поле вы видите.



Добавление CheckBox

1 В окне **Toolbox** нажмите Checkbox.

2 При выбранной кнопке Checkbox поместите курсор (указатель мыши) над пустой формой.

3 Щелкните где-нибудь в середине формы ниже других элементов управления. Обратите внимание, что там, где вы нажали, будет находиться верхний левый угол выключателя.

4 Подобно текстовой зоне элемент управления “выключатель” должен быть связан с полем или другим выражением. Однако чтобы выключатель работал корректно, выражение или поле, с которым он связан, должны возвратить булево значение (true – истину или false – ложь). Флажок в выключателе обозначает истину, а пустая рамка - ложь.

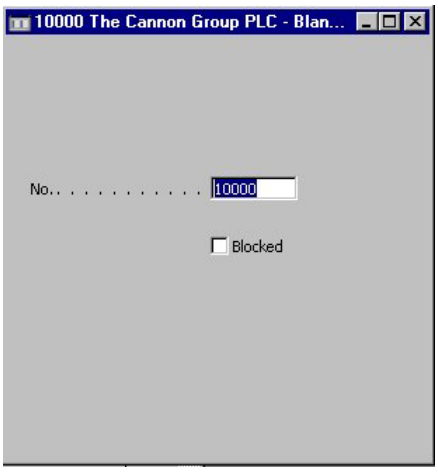
5 Откройте окно **Properties**.

6 Измените свойство SourceExpr на *Blocked*.

7 Выключатель имеет встроенную надпись. Поэтому измените значение свойства Caption на *Blocked*.

8 Вам придется изменить значение свойства ShowCaption на *Yes* и возможно увеличить размер выключателя, чтобы видеть заголовок. Обратите внимание, что встроенная надпись для выключателя находится справа от него. Это можно изменить с помощью другого свойства (вы можете поэкспериментировать, чтобы выяснить, какого именно).

9 Закройте, сохраните и откомпилируйте форму.



10 Откройте форму и проверьте новый выключатель, перемещаясь в

следующую, предыдущую, первую или последнюю записи. Кроме того, вам нужно поставить и убрать флажок.

Упражнение 3.2.4 Использование опций Add Label и Lock

Есть две очень полезных опции у основания окна **Toolbox**. Они помогут вам быстро конструировать формы. Первая – это опция Add Label (“Добавить надпись”). Она позволяет вам автоматически присоединять надпись, когда вы помещаете другой тип элемента управления в форму, такой как текстовая зона. Вторая опция - Lock. Она дает возможность одновременно добавлять несколько элементов управления.

Использование опции Add Label

1 Откройте форму 91100 (Blank Form) в Дизайнере форм.

2 Нажмите кнопку Toolbox на панели инструментов (или View, Toolbox). Появится окно **Toolbox**.

3 Нажмите кнопку Add Label (справа внизу). Обратите внимание, что эта кнопка остается нажатой. Чтобы отменить ее выбор (вернуть в ненажатое положение), вы можете снова щелкнуть ее. Она будет оставаться нажатой до тех пор, пока вы не отмените ее выбор или не выйдите из Дизайнера форм.

4 Нажмите кнопку Text Box. Обратите внимание, что кнопка Add Label все еще выбрана.

5 Поместите курсор над формой (ниже других элементов управления) и щелкните, чтобы поместить текстовую зону на форму.

Что еще было помещено на форму?

Что происходит, когда вы перемещаете текстовую зону?

Что происходит, когда вы перемещаете надпись?

Как можно переместить только текстовую зону (подсказка: ищите курсор в форме руки)?

Что происходит, если вы нажимаете кнопку Checkbox, а затем щелкаете форму?

6 Удалите элементы управления, которые вы добавили при выполнении этого упражнения.

7 Закройте, сохраните и откомпилируйте форму.

Использование опции Lock

1 Откройте форму 91100 (Blank Form) в Дизайнере форм.

2 Откройте Toolbox.

3 Нажмите кнопку Lock (слева внизу). Обратите внимание, что эта кнопка остается нажатой. Чтобы отменить ее выбор (вернуть в ненажатое положение), вы можете снова щелкнуть ее. Она будет оставаться нажатой до тех пор, пока вы не отмените ее выбор или не выйдите из Дизайнера форм.

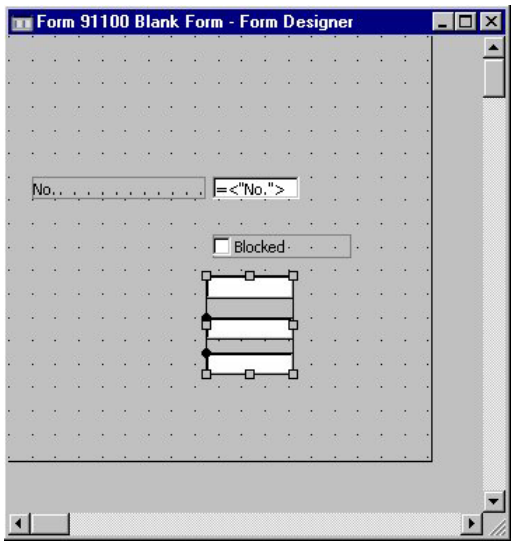
4 Нажмите кнопку Text Box. Обратите внимание, что кнопка Lock все еще выбрана.

5 Поместите курсор над формой (ниже других элементов управления) и щелкните, чтобы поместить текстовую зону на форму.

6 Переместите курсор вниз и щелкните снова.

7 Переместите курсор вниз и щелкните снова.

8 Переместите курсор вниз и щелкните снова.



Сколько текстовых зон вы добавили?

Можете ли вы перемещать какие-либо элементы управления при выбранной кнопке Lock?

9 Удалите элементы управления, которые вы добавили при выполнении этого упражнения, выбирая их по одному и нажимая клавишу DELETE или используя Marquee, чтобы выбрать их все, и затем нажмите клавишу DELETE. За более подробной информацией о выборе элементов управления обращайтесь к *«Руководству разработчика приложений»* (*Application Designer's Guide).*

10 Закройте, сохраните и откомпилируйте форму.

Использование опций Add Label и Lock

Сконструируйте форму еще раз и используйте кнопки Add Label и Lock для добавления текстовых зон. Поэкспериментируйте с обеими кнопками до тех пор, пока не освоите эти действия.

Не забудьте удалить все несвязанные элементы управления перед тем, как сохранять форму.

Многоязычные функциональные возможности

При добавлении несвязанного элемента управления на форму заполните свойство CaptionML так, как описано ранее в главе “Основные таблицы”. Если новый элемент управления – кнопка Option, то заполните и OptionCaptionML. Если вы создали новое поле в таблице и добавляете его к форме, CaptionML не станет отображаться до тех пор, пока форма не будет откомпилирована. Для New Forms потребуется заполнить и Form CaptionML.

3.3 Другие полезные элементы управления

В этом разделе вы будете создавать и использовать различные типы элементов управления. Начнем опять-таки с заведения пустой формы.

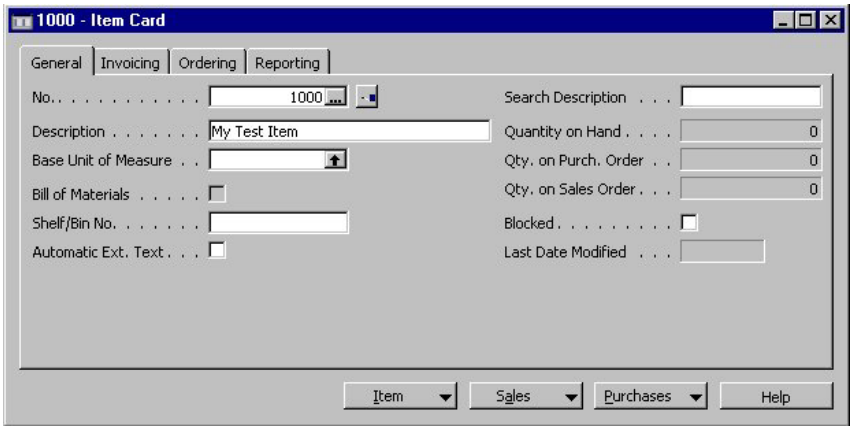
При выполнении некоторых из этих упражнений вам потребуется создать новую запись. Для создания этой записи выполните следующие действия.

1 Откройте окно **Item Card** из меню Inventory.

2 Нажмите клавишу F3, чтобы вставить новую запись.

3 В поле **No.** введите *1111*.

4 В поле **Description** введите *My Test Item*.



5 Закройте форму **Item Card**.

6 Получив инструкцию найти элемент 1000, вы должны найти элемент, который только что создали.

Упражнение 3.3.1 Добавление контейнерных элементов управления

В этом упражнении вы добавите контейнерные элементы управления к пустой форме. Вы будете использовать такие элементы управления, как закладка (tab), фрейм (frame) и табличная зона (table box).

1 Создайте пустую форму так, как это было сделано в упражнении 3.2.1. Введите имя «Blank Form 2» и идентификатор 91200.

2 Выполните действия, описанные в упражнении 3.2.2, чтобы установить SourceTable на Item (таблица **Customer** не может использоваться в некоторых упражнениях).

3 Откройте форму 91200 в Дизайнере форм.

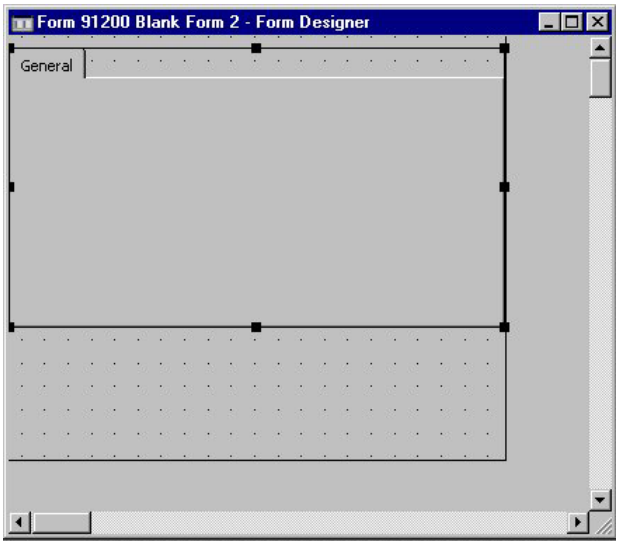
Использование закладки

1 Откройте Toolbox.

2 Нажмите кнопку Tab Control.

3 Поместите курсор над левым верхним краем формы.

4 Щелкните, чтобы создать закладку. Обратите внимание, что он содержит только одну вкладку. Имя первой вкладки – General. Это имя стандартное для форм, содержащих закладки.



Чтобы добавить другие вкладки к элементу управления “закладка”, выполните следующие действия.

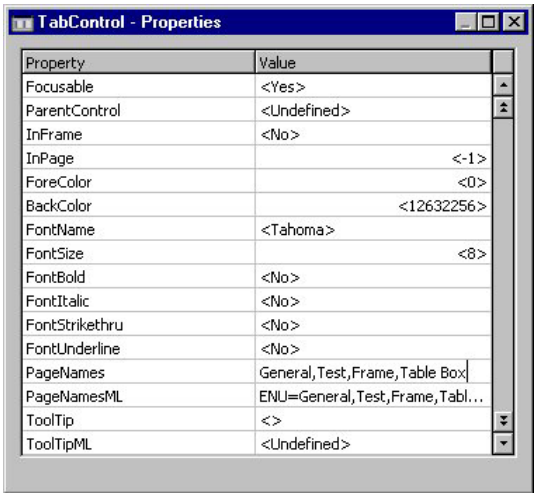
1 Щелкните первую вкладку на элементе управления “закладка” (подойдет любая). Это может привести к выбору элемента управления “закладка” или просто к переходу на эту вкладку.

2 Чтобы убедиться в выборе закладки, снова щелкните первую вкладку.

3 При выбранной закладке откройте окно **Properties**.

4 Щелкните свойство PageNames. Обратите внимание, что текущее значение - <General> (имя первой и единственной вкладки). Свойство PageNames - отделенный запятыми список имен вкладок. Для добавления вкладки вы просто добавляете ее к списку. (Вы можете дважды нажать функциональную клавишу F2, чтобы перейти в конец выбранного поля.)

5 Измените свойство PageNames, введя *General,Test,Frame,Table Box*. После запятых пробелов не должно быть.



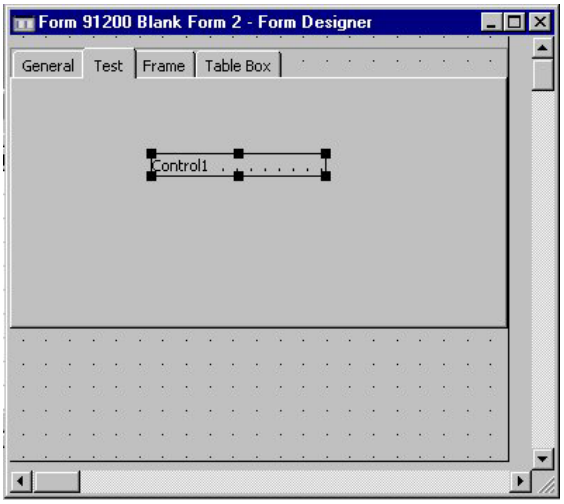
Что произошло с элементом управления “закладка”?

Добавление элементов управления на вкладку

1 Щелкните закладку, к которой вы хотите добавить элемент управления. В этом случае нажмите вкладку Test.

2 Выберите элемент управления в окне Toolbox. Выберите элемент управления “надпись”.

3 Щелкните Tab там, где вы хотите поместить элемент управления. Вы можете при необходимости переместить элемент управления.



Что происходит, если вы нажимаете на вкладку General и затем снова на вкладку Test?

4 Отредактируйте свойство PageNames элемента управления “закладка” так, чтобы он выглядел следующим образом: General, Frame, Table Box. Где теперь находится элемент управления “надпись”?

5 Переместите надпись в нижнюю часть формы (в сторону от закладки). Что происходит, когда вы переходите на вкладку General и снова возвращаетесь назад?

6 Нажмите вкладку General и переместите надпись из основания формы в середину страницы для вкладки General. Что происходит, когда вы переходите на вкладку Frame и снова возвращаетесь назад?

Добавление фрейма

Элемент управления “фрейм” подобен странице элемента управления “закладка”. Он содержит другие элементы управления, и когда вы перемещаете фрейм, эти элементы управления перемещаются вместе с ним.

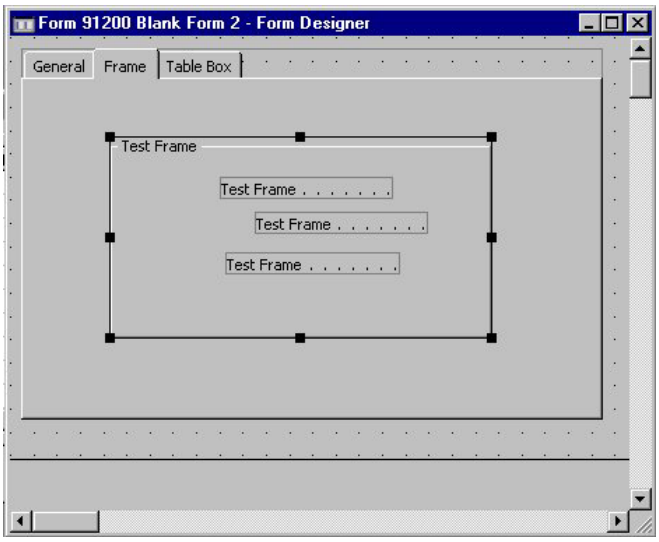
1 Нажмите вкладку **Frame** элемента управления “закладка”.

2 В окне **Toolbox** выберите кнопку Frame.

3 Щелкните в середине страницы для создания вкладки **Frame**. Должен быть создан фрейм.

4 Измените свойство Caption фрейма на *Test Frame*.

5 Добавьте к фрейму несколько надписей. Не забудьте установить элементы управления внутрь фрейма.



Что происходит, когда вы перемещаете фрейм?

Что происходит, когда вы удаляете фрейм?

Добавление табличной зоны

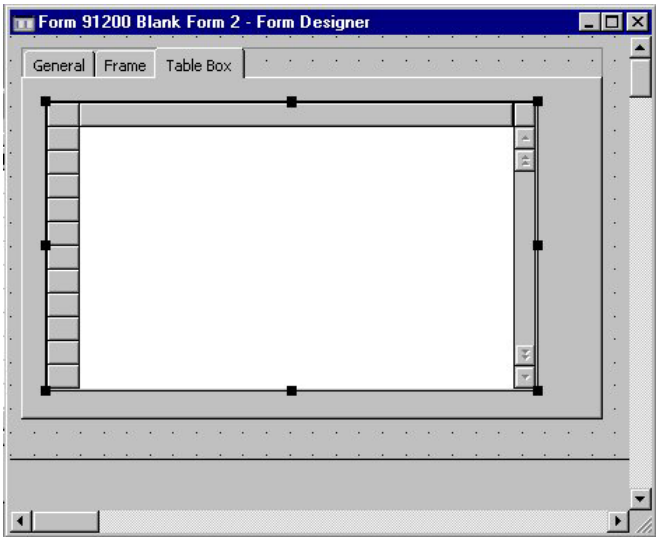
Табличная зона – это контейнерный элемент управления, который содержит связанные элементы управления. Он позволяет отображать на форме более одной записи одновременно.

1 Нажмите вкладку **Table Box** элемента управления “закладка”.

2 В окне **Toolbox** выберите кнопку TableBox.

3 Щелкните в середине вкладки, чтобы создать табличную зону. Возможно, вам потребуется переместить ее в верхний левый угол. Обратите внимание, что табличная зона пуста. Помните, что это контейнерный элемент управления. Необходимо добавить к нему другие элементы управления.

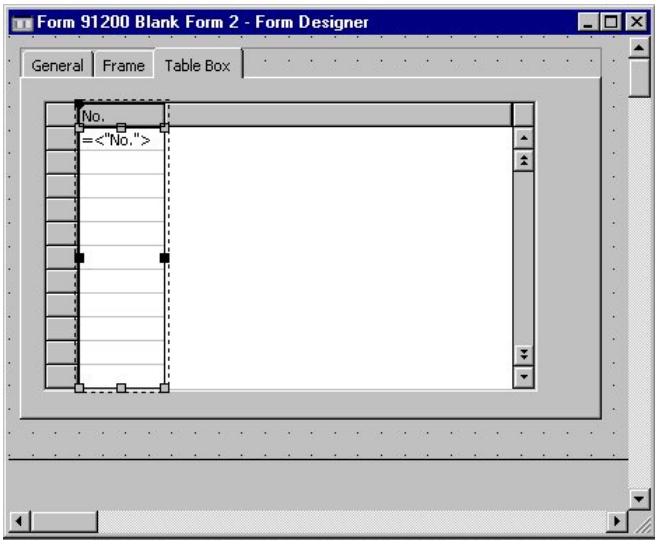
Чаще всего в табличной зоне можно увидеть такой элемент управления, как текстовая зона.



4 В окне Toolbox выберите кнопку Text Box.

5 Щелкните TableBox. Текстовая зона становится столбцом, а элемент управления “метка” добавляется в верхнюю часть для указания заголовка столбца.

6 Измените свойство SourceExpr элемента управления “текстовая зона” на *«”No”»* (включая двойные кавычки).



7 Закройте, сохраните и откомпилируйте форму.

8 Откройте форму. Что отображается в табличной зоне?

Что происходит со строкой заголовка формы, если вы щелкаете в другой строке табличной зоны?

Что происходит, если вы разворачиваете форму или просто увеличиваете ее?

Элементы управления для автоматической регулировки размера

Было бы лучше, если бы закладка увеличивалась вместе с формой. Табличная форма тоже должна увеличиваться вместе с формой. Чтобы добиться этого, мы откорректируем связующие свойства элементов управления.

1 Откройте форму 91200 в Дизайнере форм.

2 Выберите элемент управления “закладка” двойным щелчком по вкладке General.

3 Откройте окно **Properties**.

4 Измените значение свойства HorzGlue на *Both*.

5 Измените значение свойства VertGlue на *Both*.

6 Выберите табличную зону, нажав вкладку TableBox и затем щелкнув верхний левый серый прямоугольник в табличной зоне.

7 Откройте окно **Properties**.

8 Измените значение свойства HorzGlue на *Both*.

9 Измените значение свойства VertGlue на *Both*.

10 Закройте, сохраните и откомпилируйте форму.

11 Откройте форму. Что происходит теперь, когда вы изменяете размер формы, делая ее больше?

Упражнение 3.3.2 Опции отображения

Чтобы пользователь видел дополнительные поля, их можно отобразить двумя различными способами. Вы можете использовать текстовую зону (которая может отображать почти все) или кнопки

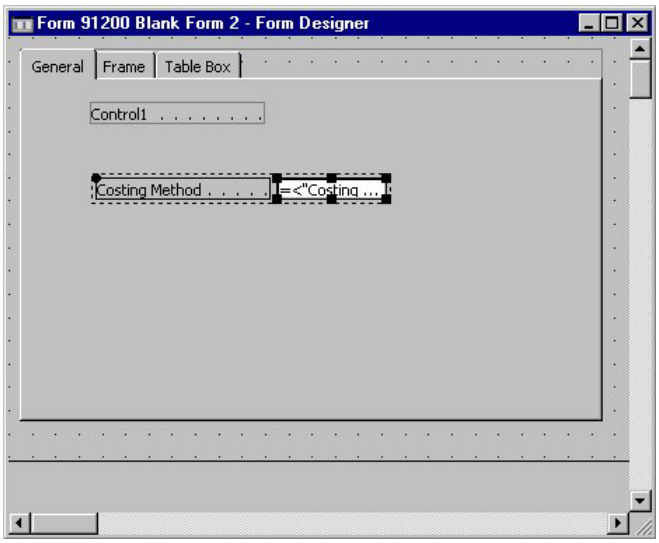
выбора вариантов (также называемые «переключателями» – radio buttons). В этом упражнении вы изучите оба способа.

Использование текстовой зоны

1 Откройте форму 91200 в Дизайнере форм.

2 Добавьте текстовую зону с соответствующей надписью на вкладку **General**.

3 Измените значение свойства SourceExpr текстовой зоны на *Costing Method*. Поле **Costing Method** – это необязательное поле в таблице **Item**.



4 Закройте, сохраните и откомпилируйте форму.

5 Откройте форму.

Какая кнопка находится в текстовой зоне?

Что происходит, когда вы нажимаете кнопку?

Использование кнопок выбора

1 Откройте форму 91200 в Дизайнере форм.

2 В окне **Toolbox** нажмите кнопку Lock, чтобы выбрать ее.

3 В окне **Toolbox** выберите кнопку Option.

4 Щелкните вкладку **General**, чтобы создать кнопку выбора.

5 Переместите курсор вниз и щелкните снова, чтобы создать другую кнопку выбора.

6 Переместите курсор вниз и щелкните снова, чтобы создать последнюю кнопку выбора. Существует пять вариантных значений для поля **Costing Method**. Вы дадите пользователю возможность обращаться только к трем из этих значений. Поэтому вам нужно создать три кнопки выбора. Каждая кнопка выбора представляет одно значение поля **Costing Method**.

7 В окне **Toolbox** нажмите кнопку Lock, чтобы отменить ее выбор.

8 Для каждой кнопки выбора измените значение свойства SourceExpr на «Costing Method».

9 Для первой кнопки выбора измените значение свойства OptionValue на 0.

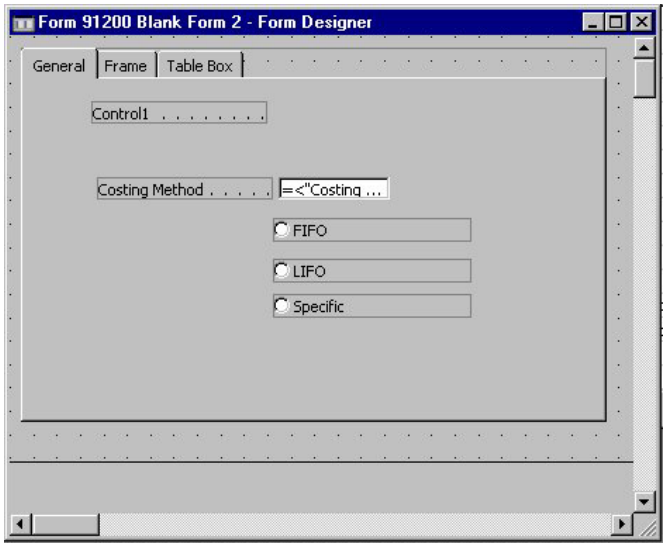
10 Для первой кнопки выбора измените значение свойства Caption на *FIFO.*

11 Для второй кнопки выбора измените значение свойства OptionValue на 1.

12 Для второй кнопки выбора измените значение свойства Caption на *LIFO.*

13 Для третьей кнопки выбора измените значение свойства OptionValue на 2.

14 Для третьей кнопки выбора измените значение свойства Caption на *Specific.*



15 Закройте, сохраните и откомпилируйте форму.

16 Откройте форму и найдите элемент 1000.

Сколько кнопок выбора отобрано?

Что происходит, если вы пытаетесь отобрать еще одну?

Что происходит, если вы изменяете текстовую зону, которая связана с полем Costing Method, на Specific?

Что происходит, если вы изменяете текстовую зону, которая связана с полем Costing Method, на Average?

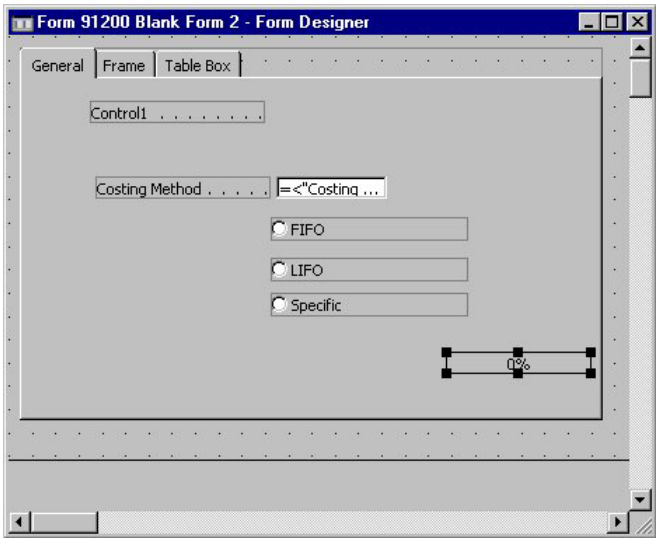
Упражнение 3.3.3 Отображение индикаторов выполнения

В этом упражнении вы создадите способ, благодаря которому пользователь легко узнает процент от прибыли, которую он получает с каждого товара. Вы будете использовать вертикальный индикатор выполнения с диапазоном значений от 0 до 100.

1 Откройте форму 91200 в Дизайнере форм.

2 В окне **Toolbox** выберите кнопку Indicator.

3 Нажмите вкладку **General**, чтобы создать индикатор. По умолчанию индикатор начинает отсчет слева направо. Ваш индикатор начнет отсчет снизу с нуля и поднимется наверх (100).



4 Измените свойство Orientation у индикатора на Bottom.

5 Измените значение свойства MinValue на *0*.

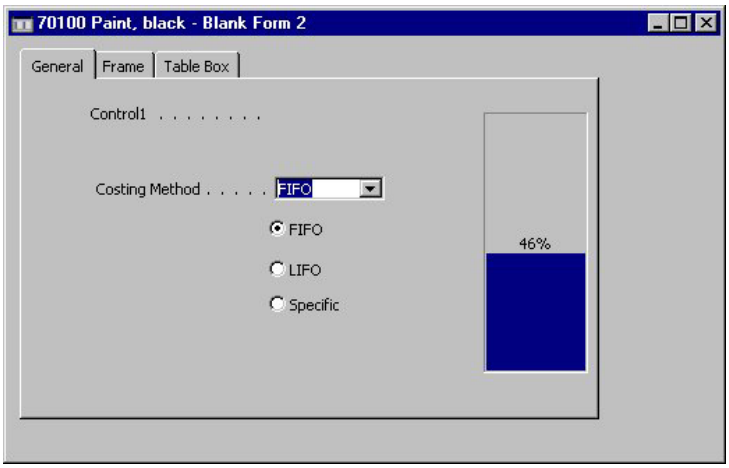
6 Измените значение свойства MinValue на *100*.

7 Измените значение свойства SourceExpr на «*Profit %*».

8 Увеличьте высоту индикатора, перемещая верхний край.

9 Закройте, сохраните и откомпилируйте форму.

10 Откройте форму.



Используйте кнопку Next, чтобы просмотреть несколько различных записей. Какой ряд значений вы увидели?

Что происходит, если вы изменяете значение свойства MaxValue на *1000* и снова открываете форму?

Упражнение 3.3.4 Отображение BLOB-полей (рисунков)

BLOB-поля используются в основном для хранения изображений. Некоторые элементы в таблице Item содержат изображения. Вы добавите элемент управления в свою форму, чтобы показать изображение пользователю.

1 Откройте форму 91200 в Дизайнере форм.

2 Выберите элемент управления “закладка” двойным щелчком по вкладке General.

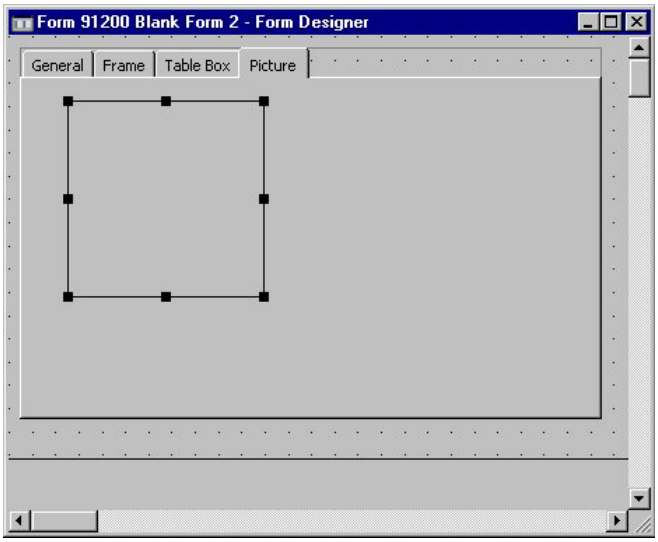
3 Добавьте вкладку к элементу управления “закладка” с именем Picture. Убедитесь, что это последняя вкладка на элементе управления.

4 Нажмите вкладку Picture.

5 В окне **Toolbox** нажмите кнопку PictureBox.

6 Щелкните в середине вкладки, чтобы создать зону изображения. Не беспокойтесь о размере. Заданный по умолчанию размер вполне подходит для этого примера.

7 Измените значение свойства SourceExpr у зоны изображения на *Picture*.



8 Закройте, сохраните и откомпилируйте форму.

9 Откройте форму.

Используйте кнопку Next, чтобы просмотреть несколько различных записей. Перечислите некоторые элементы, содержащие изображения.

3.4 Использование мастера форм

В этом разделе вы изучите более простые способы создания и изменения форм. Всегда имейте в виду то, что вы узнали в предыдущих разделах, чтобы понимать, какие выполняются действия.

Мастер формы может создавать два различных типа форм – картотечные формы (Card) и табличные формы (Tabular).

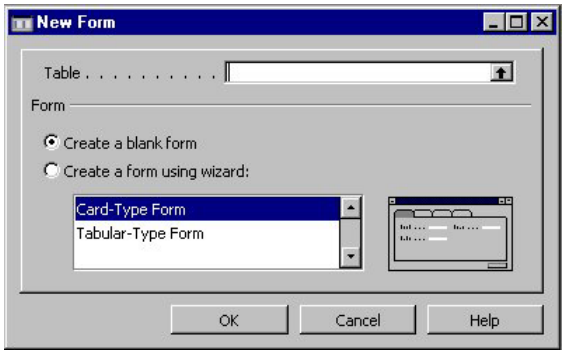
Упражнение 3.4.1 Картотечные формы

Картотечная форма– это форма, которая отображает данные одной записи. Обычно она содержит закладку, а первая вкладка обычно называется «General». В этом упражнении вы создадите картотечную форму с помощью соответствующего мастера.

Запуск мастера

1 В Дизайнере объектов нажмите кнопку Form.

2 Нажмите кнопку New, и приложение C/SIDE отобразит окно **New Form**.



3 Введите число 18 в поле Table. Вы создаете новую форму **Customer Card**.

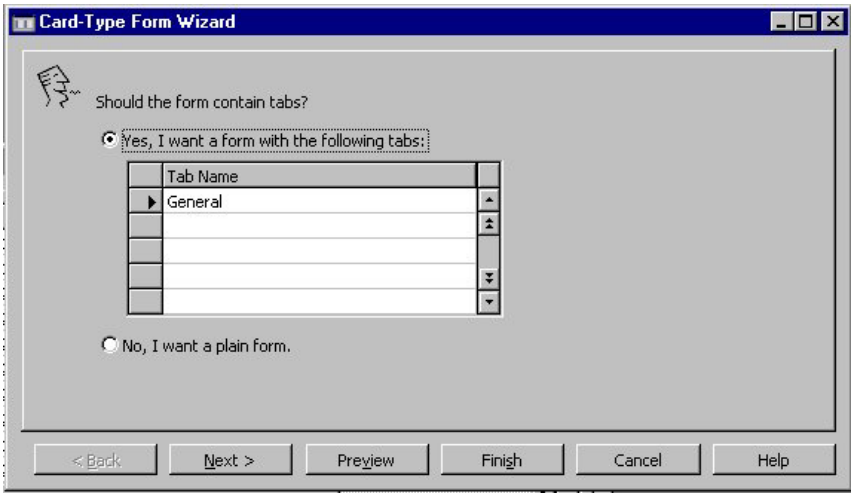
4 Щелкните «Create a form using wizard».

5 Щелкните «Card-Type Form» в списке.

6 Нажмите кнопку OK, чтобы запустить мастера картотечных форм.

Мастер картотечных форм

В любой момент во время создания формы в мастере вы можете нажать кнопку Preview в нижней части окна мастера, чтобы увидеть то, что вы создаете.



1 Должна ли форма содержать закладки?

2 Пусть переключатель Yes останется включенным. Обратите внимание, что первая вкладка по умолчанию называется General.

3 Щелкните в пустой строке под General и введите *Second Tab*.

4 Нажмите кнопку Next.

Какие поля из таблицы **Customer** вы хотите поместить на вкладки?

5 Нажмите вкладку **Second Tab**. Обратите внимание, что форма содержит обе вкладки, которые вы запросили.

6 Щелкните вкладку **General**.

7 Выполните двойной щелчок на следующих полях: **No.**, **Name**, **Address** и **City**. Эти поля перемещены на вкладку **General**.

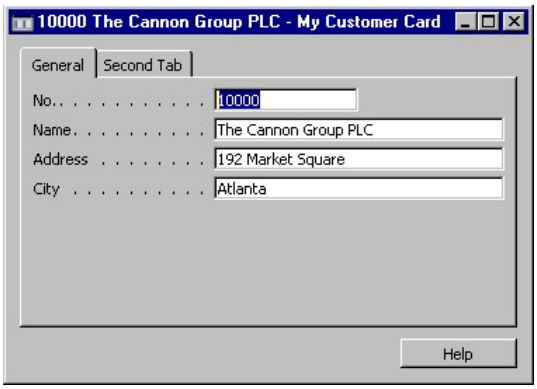
8 Нажмите вкладку **Second Tab**. Обратите внимание, что список Field Order пуст для этой вкладки.

9 Выполните двойной щелчок на следующих полях: **Global Dimension 1 Code**, **Territory Code**, **Global Dimension 2 Code**.

10 Нажмите кнопку Finish, чтобы завершить работу мастера, и откройте только что созданную форму в Дизайнере форм. Форма еще не сохранена.

11 Закройте, сохраните и откомпилируйте форму с идентификатором 91300 и именем **My Customer Card**.

12 Откройте форму и проверьте вкладки и поля.



Упражнение 3.4.2 Табличные формы

Табличная форма – это форма, которая отображает одновременно данные нескольких записей. Обычно она содержит табличную зону. В этом упражнении вы создадите табличную форму с помощью соответствующего мастера.

Запуск мастера

1 В Дизайнере объектов нажмите кнопку Form.

2 Нажмите кнопку New. Приложение C/SIDE отобразит окно **New Form**.

3 Введите число *18* в поле Table. Вы создаете новую форму **Customer List**.

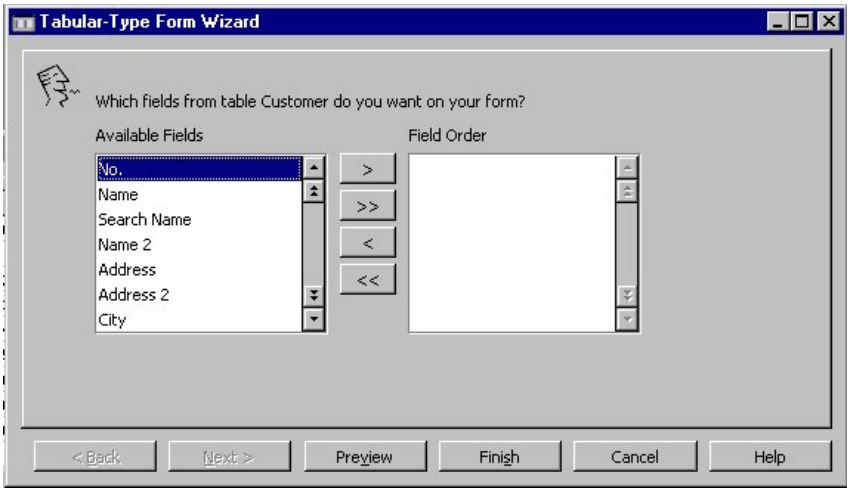
4 Щелкните «Create a form using wizard».

5 Щелкните «Tabular-Type Form» в списке.

6 Нажмите кнопку OK, чтобы запустить мастер табличных форм.

Мастер табличных форм

Чтобы увидеть то, что вы создаете, в любой момент во время создания формы в мастере можно нажать кнопку Preview в нижней части окна мастера. Обратите внимание, что в этом мастере есть только одно окно. Закончив работу в этом окне, вы завершаете работу мастера.



1 Какие поля из таблицы **Customer** вы хотите поместить в свою форму?

2 Выполните двойной щелчок на следующих полях: **No.**, **Name**, **Address** и **City**.

3 Нажмите кнопку Finish, чтобы завершить работу мастера, и откройте только что созданную форму в Дизайнере форм. Форма еще не сохранена.

4 Закройте, сохраните и откомпилируйте форму с идентификатором 91301 и именем **My Customer List**.

5 Откройте форму и проверьте поля.

No.	Name	Address	City
10000	The Cannon Group PLC	192 Market Square	Atlanta
20000	James Dailey Ltd.	153 Thomas Drive	Atlanta
30000	John Haddock Insurance Co.	10 High Tower Green	Atlanta
40000	Deerfield Graphics Company	10 Deerfield Road	Miami
50000	Guildford Water Department	25 Water Way	Chicago
01121212	Spotsmeyer's Furnishings	612 South Sunset Drive	Miami
01445544	Progressive Home Furnishings	3000 Roosevelt Blvd.	Chicago
01454545	New Concepts Furniture	705 West Peachtree Street	Atlanta
31505050	Woonboulevard Kuitenbrouwer	Industrieweg 11	
31669966	Meersen Meubelen	Vijfpoortenweg 71	
31987987	Jean de Bouvier	Westzijdewal 123	

В отличие от настоящей формы **Customer List**, эта форма позволяет пользователю изменять значения. Это нежелательно, потому что форма списка используется для поиска и выбора записей. Вы должны изменить форму так, чтобы она стала недоступной для редактирования. Вы можете изменить все элементы управления, но выполнение удаления и вставки все равно будет возможно.

1 Откройте форму 91301 в Дизайнере форм.

2 Откройте окно **Properties** этой формы.

3 Измените значение свойства Editable формы на *No*.

4 Закройте, сохраните и откомпилируйте форму.

5 Откройте форму.

6 Щелкните поле **Name** и введите свое имя.

Что происходит?

Почему это желательно пользователю?

Упражнение 3.4.3 Использование меню полей

В этом упражнении вы добавите поля к своей картотечной форме и к списочной форме.

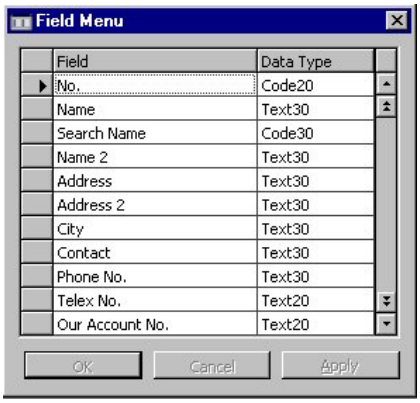
Примечание

Меню полей (Field Menu) можно использовать для добавления полей к любой связанной форме независимо от того, использовали вы мастер для создания формы или нет.

Добавление полей к картотечной форме

1 Откройте форму 91300 в Дизайнере форм.

2 На панели инструментов нажмите кнопку Field Menu, чтобы открыть окно **Field Menu**.



3 Найдите поле **Balance** в окне **Field Menu**.

4 Щелкните серый прямоугольник около поля **Balance**, чтобы выбрать поле (оно должно быть выделено синим цветом).

5 Переместите мышь на форму, где вы хотите поместить поле.

6 Сделайте щелчок, чтобы создать текстовую зону для этого поля. Свойство SourceExpr уже установлено.

7 Закройте, сохраните и откомпилируйте форму.

8 Откройте форму.

Работает ли поле должным образом?

Что еще помещено на форму вместе с текстовой зоной?

Добавление полей на табличную форму

1 Откройте форму 91301 в Дизайнере форм.

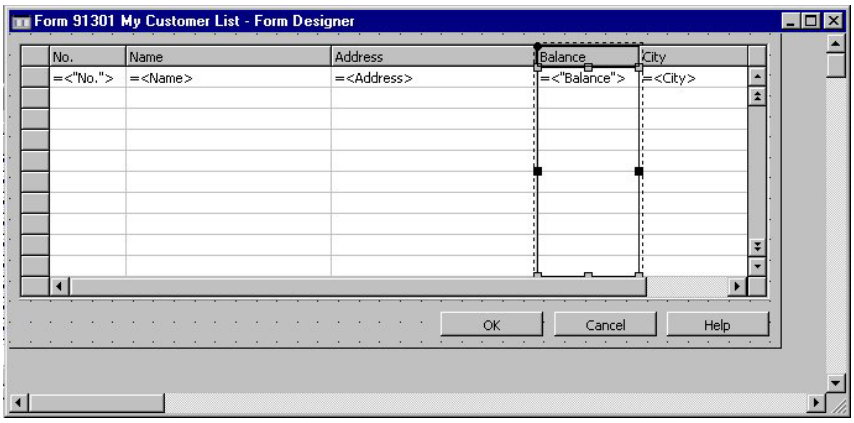
2 На панели инструментов нажмите кнопку Field Menu, чтобы открыть окно **Field Menu**.

3 Найдите поле **Balance** в окне **Field Menu**.

4 Щелкните серый прямоугольник около поля **Balance**, чтобы выбрать поле (оно должно быть выделено синим цветом).

5 Переместите мышь на табличную зону, где вы хотите поместить новый столбец.

6 Сделайте щелчок, чтобы создать новый столбец для этого поля. Свойство SourceExpr уже установлено.



7 Закройте, сохраните и откомпилируйте форму.

8 Откройте форму.

Работает ли поле должным образом?

Каков был заголовок столбца?

Считаете ли вы, что большинству пользователей необходимо видеть поле Balance на этой форме?

Упражнение 3.4.4 Как сделать невидимыми столбцы на табличной форме

Поскольку большинству пользователей не нужно видеть баланс клиента, столбец на форме **List** необходимо сделать невидимым по умолчанию. Однако каждый пользователь должен иметь возможность добавить его.

В этом упражнении вы измените форму **List** так, чтобы она соответствовала указанным выше требованиям.

1 Откройте форму 91301 в Дизайнере форм.

2 Откройте окно **Properties** для отображения столбца поля **Balance**.

3 Измените значение свойства Visible у текстовой зоны на *No*.

4 Закройте, сохраните и откомпилируйте форму.

5 Откройте форму.

Видите ли вы столбец?

6 Щелкните View, Show Column. Находится ли поле **Balance** в списке? Находится ли поле **Net Change** в списке?

Почему поле **Balance** внесено в список, а поле **Net Change** – нет?

7 Поставьте флажок рядом с полем **Balance** и нажмите кнопку OK. Отображается ли поле теперь **Balance**?

Что происходит, если вы закрываете форму и открываете ее снова?

3.5 Создание кнопок

В этом разделе вы узнаете о наиболее важном элементе управления, который вам придется добавлять почти во все формы (мастер не будет делать этого за вас). Этот элемент управления – кнопка (button).

Вы изучите два основных типа кнопок – командные кнопки (command buttons) и кнопки меню (menu buttons).

Упражнение 3.5.1 Создание командных кнопок

В этом упражнении вы создадите командную кнопку, которая закрывает форму.

1 Откройте форму 91300 в Дизайнере форм.

2 Откройте окно **Toolbox**.

3 Нажмите Command Button.

4 Поместите курсор мыши в нижней правой части формы (слева от кнопки Help).

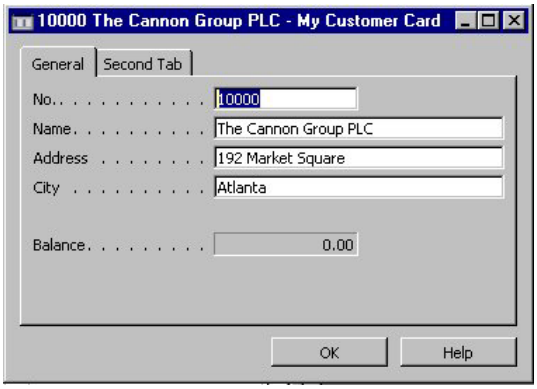
5 Щелкните, чтобы создать командную кнопку. В стандартных формах кнопки помещаются в нижний правый угол.

6 Откройте окно **Properties** для отображения свойств этой кнопки.

7 Измените значение свойства PushAction кнопки на *OK* (но не LookupOK). Благодаря этому данная кнопка будет кнопкой OK. При нажатии кнопки OK закрывают форму. Свойство PushAction должно быть установлено для любой кнопки, для которой вы не будете писать программный код.

8 Закройте, сохраните и откомпилируйте форму.

9 Откройте форму.



Какой заголовок имеет кнопка?

Что происходит, когда вы нажимаете кнопку?

10 Снова откройте форму. Медленно измените размеры формы. Что происходит с вашей новой кнопкой?

Что произошло с кнопкой Help?

Что заставило их действовать по-разному?

11 Измените свойства HorzGlue и VertGlue своей кнопки так, чтобы они соответствовали свойствам кнопки Help.

12 Снова протестируйте форму, изменив ее размеры.

Что происходит с вашей новой кнопкой?

Упражнение 3.5.2 Значение RunObject свойства PushAction

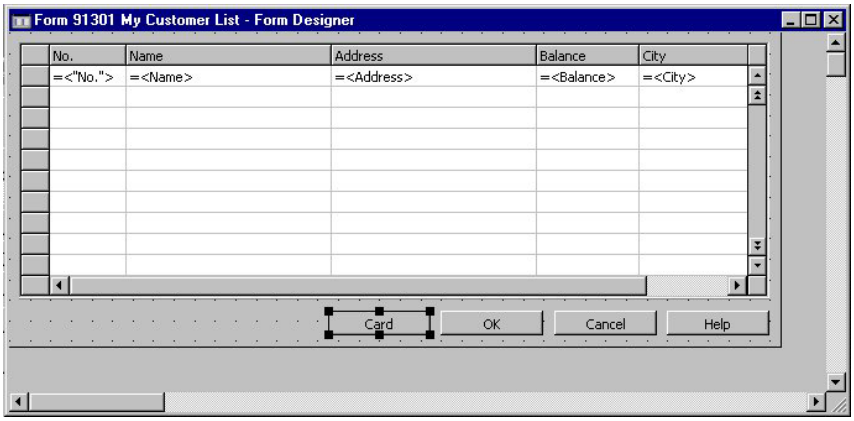
В этом упражнении вы создадите кнопку, которая открывает другую форму.

1 Откройте форму 91301 в Дизайнере форм.

2 Добавьте командную кнопку (см. предыдущее упражнение) рядом с кнопкой OK.

3 Измените заголовок кнопки на *«Card»*. Не забудьте изменить и связующие свойства.

4 Измените значение свойства RunObject на «*form 91300*» (без кавычек). Кроме того, вы можете использовать кнопку поиска, чтобы найти форму 91300 Ввод значений обычно происходит быстрее.



5 Закройте, сохраните и откомпилируйте форму.

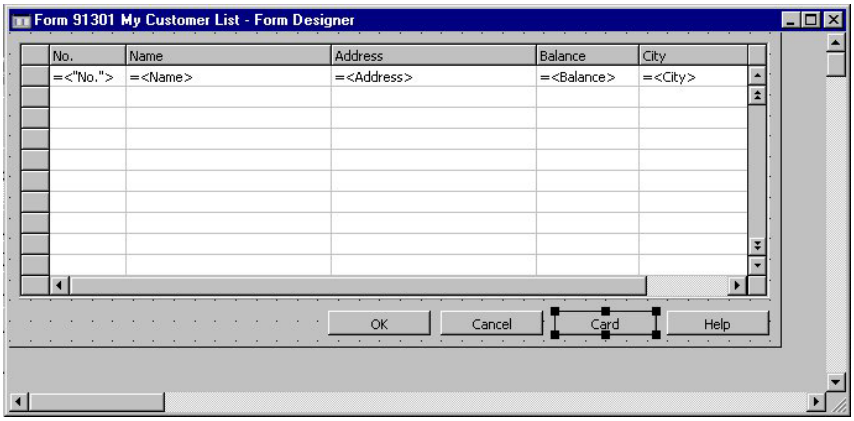
6 Откройте форму.

Где находится кнопка? Где находятся кнопки OK и Cancel?

Что происходит, когда вы нажимаете кнопку?

Что вы не установили?

7 Измените значение свойства PushAction кнопки на *RunObject*. Теперь кнопка выполнит действие. PushAction <0> заставит кнопку не производить никаких действий (кроме запуска своего триггера OnPush). Возможно, вам также потребуется переместить свою картотечную кнопку, поставив ее рядом с кнопкой Help.



8 Снова откройте форму. Что происходит на этот раз при нажатии кнопки Card?

9 Закройте картотечную форму. Выберите клиента 50000 из списочной формы. Нажмите кнопку Card. Какой клиент отображен на картотечной форме?

Упражнение 3.5.3 Формы для связывания

В этом упражнении вы измените кнопку Card так, чтобы она открыла картотечную форму с записью, которая выбрана в списочной форме. Это называется связыванием (linking).

1 Откройте форму 91301 в Дизайнере форм.

2 Откройте окно **Properties** для отображения свойств кнопки Card. Прямо под свойством RunObject находятся свойства RunForm. Эти свойства позволяют вам управлять формой, которую открывает кнопка.

3 Измените значение свойства RunFormLink на *No.=FIELD(No.)*. Вы можете также использовать кнопку поясняющего редактирования (assist-edit), чтобы вызвать диалог Table Filter, который вам поможет.

4 Закройте, сохраните и откомпилируйте форму.

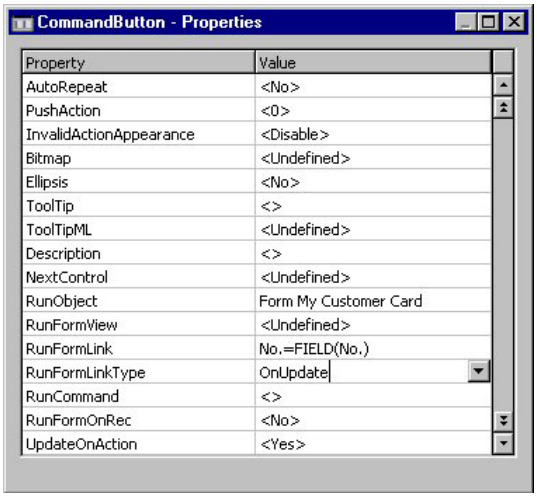
5 Откройте форму.

Что происходит, когда вы нажимаете кнопку Card при условии, что выбран клиент 50000?

Что происходит, если вы закрываете картотечную форму, выбираете другого клиента и снова нажимаете кнопку Card?

Обратите внимание, что пока картотечная форма открыта, вы можете снова щелкнуть списочную форму и выбрать другую запись. Что происходит с картотечной формой, когда вы изменяете списочную форму подобным образом?

6 Сконструируйте форму и измените значение свойства RunFormLinkType кнопки Card на *OnUpdate*.



7 Сохраните изменения и откройте форму. Что теперь происходит, когда вы изменяете списочную форму, пока картотечная форма открыта?

Упражнение 3.5.4 Создание кнопок меню

Кнопки меню – это контейнеры для пунктов меню (menu item). Пункт меню почти идентичен кнопке (у них много одних и тех же свойств).

В этом упражнении вы создадите кнопку меню.

1 Откройте форму 91301 в Дизайнере форм.

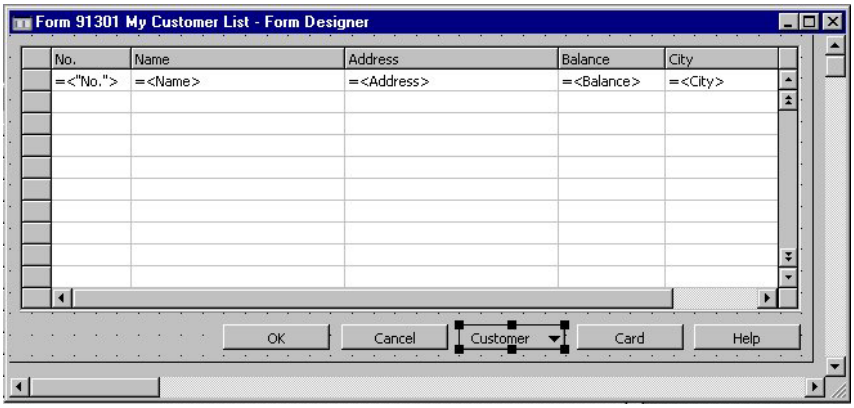
2 Переместите кнопки OK и Cancel налево.

3 В окне **Toolbox** нажмите Menu Button.

4 Щелкните форму рядом с кнопкой Card.

5 Измените заголовок кнопки меню на *«Customer».* Большинство картотечных и списочных форм содержит кнопку меню с именем таблицы.

6 Измените связующие свойства кнопки, что вы уже делали при работе с другими кнопками.



7 Закройте, сохраните и откомпилируйте форму.

8 Откройте форму.

Что происходит, когда вы нажимаете новую кнопку меню?

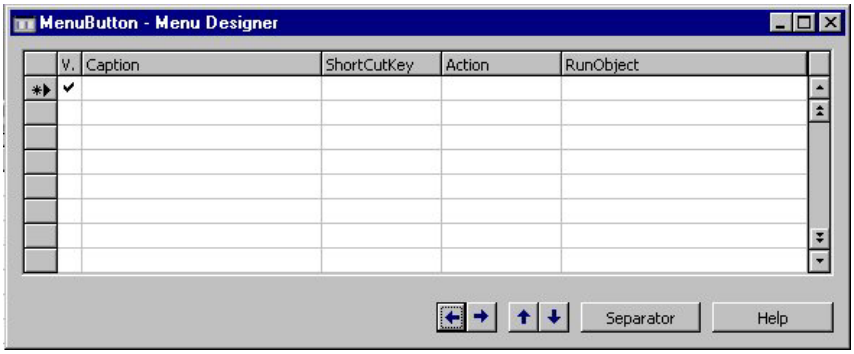
Упражнение 3.5.5 Создание пунктов меню

В этом упражнении вы добавите пункты меню к своей кнопке меню.

1 Откройте форму 91301 в Дизайнере форм.

2 Нажмите кнопку меню Customer, чтобы выбрать ее.

3 Нажмите View, Menu Items, чтобы открыть окно Menu Designer.



4 В поле **Caption**, в первой пустой строке, введите слово *Card*.

5 Вместо ввода остальных значений в строке откройте окно **Properties** для отображения пункта меню.

6 Измените значение свойства PushAction на RunObject.

7 Измените значение свойства RunObject на *91300*.

8 Измените значение свойства RunFormLink на *No.=FIELD(No.)*. Вы можете также использовать кнопку поясняющего редактирования (assist-edit), чтобы открыть окно **Filter** и установить значения там.

9 Измените значение свойства RunFormLinkType на *OnUpdate*.

10 Измените значение свойства ShortCutKey на *Shift+F5*.

11 Закройте Menu Designer.

12 Закройте, сохраните и откомпилируйте форму.

13 Откройте форму.

Что теперь происходит, когда вы нажимаете кнопку меню?

Что происходит, когда вы нажимаете пункт меню Card?

Что происходит, когда вы нажимаете комбинацию клавиш Shift+F5?

Упражнение 3.5.6 Форма поиска (LookupTable) для PushAction

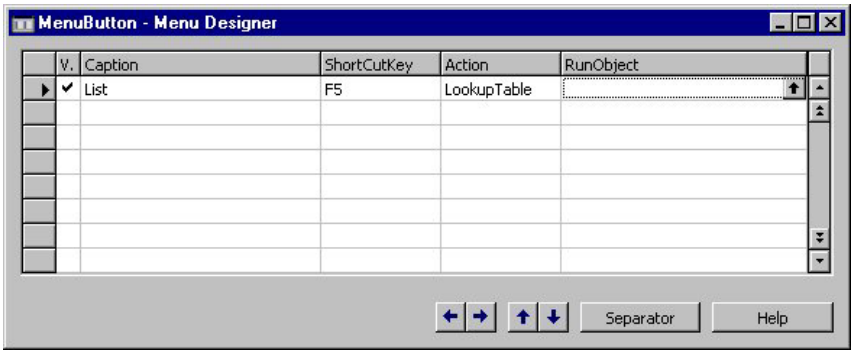
В этом упражнении вы изучите очень специфичное действие PushAction с именем LookupTable.

1 Откройте форму 91300 в Дизайнере форм.

2 Добавьте кнопку меню на форму с заголовком *Customer,* как описано в одном из предыдущих упражнений.

3 Добавьте пункт меню к кнопке меню, имеющей заголовок List.

4 Измените ShortCutKey пункта меню на F5, а PushAction – на LookupTable. Это действие вызывает заданную по умолчанию форму поиска для отображения исходной таблицы формы.



5 Закройте, сохраните и откомпилируйте форму.

6 Откройте форму.

Какая форма появляется теперь, когда вы нажимаете функциональную клавишу F5?

Почему появляется эта форма, а не ваша форма списка клиентов?

7 Откройте таблицу 18 в окне Table Designer и измените свойство LookupFormID таблицы на *91301*. Снова откройте свою картотечную форму. Какая форма появляется теперь, когда вы нажимаете функциональную клавишу F5?

3.6 СОЗДАНИЕ СОПУТСТВУЮЩИХ ФОРМ

В этом разделе вы изучите, как отображать данные из другой таблицы на форме (не из исходной таблицы формы). Один способ выполнения этого – отображение одной формы в другой. Это называется сопутствующая форма (subform). Здесь вы изучите этот элемент управления.

Вы создадите две формы, которые работают вместе, чтобы показать записи клиента и комментарии клиента на одной форме.

Упражнение 3.6.1 Создание главной формы

В этом упражнении вы создадите главную форму (main form). Она будет отображать информацию о клиенте и содержать сопутствующую форму, которая отображает комментарии, предназначенные для заказчика.

1 Откройте Дизайнер объектов и нажмите кнопку Form.

2 Нажмите кнопку New. C/SIDE отобразит окно **New Form**.

3 Создайте картотечную форму, используя мастера, который содержит номер, имя, адрес, город, страну и почтовый индекс клиента на вкладке с именем General. См. упражнение 3.3. для получения дополнительной информации об использовании мастера.

4 После завершения работы мастера по созданию формы сохраните и откомпилируйте форму с идентификатором *91500* и именем *Customer Comment*.

5 Подготовьте пространство для сопутствующей формы, увеличив размер формы и переместив кнопку Help вниз к нижнему правому углу. Для сопутствующей формы на форме должно быть почти столько же места, сколько есть для закладки.

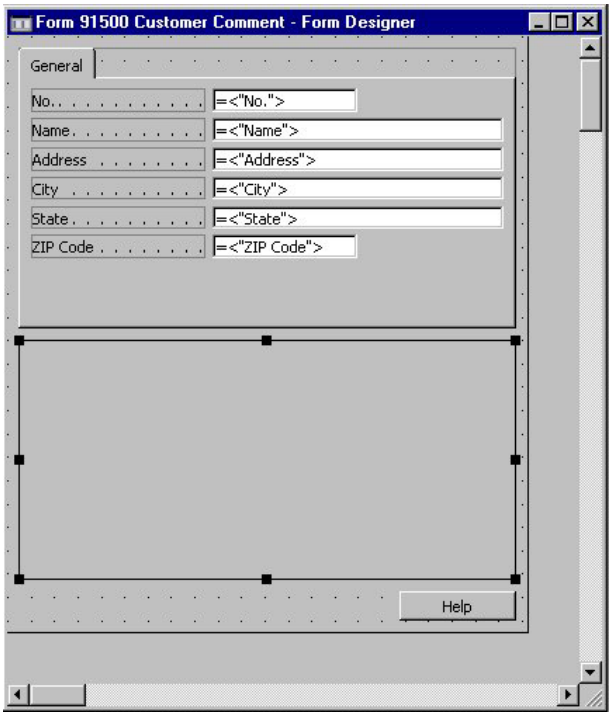
6 Создайте сопутствующую форму на форме из окна **Toolbox**. Этот элемент управления должен быть позиционирован прямо под закладкой и иметь точно такую же ширину. Откорректируйте высоту сопутствующей формы, чтобы она заняла доступное пространство.

7 Откройте окно **Properties** для отображения свойств

сопутствующей формы и запишите свойства Height (“высота”) и Width (“ширина”).

Height _____

Width _____



8 Закройте, сохраните и откомпилируйте форму.

9 Откройте форму.

Что появляется там, где находится сопутствующая форма?

Упражнение 3.6.2 Создание сопутствующей формы

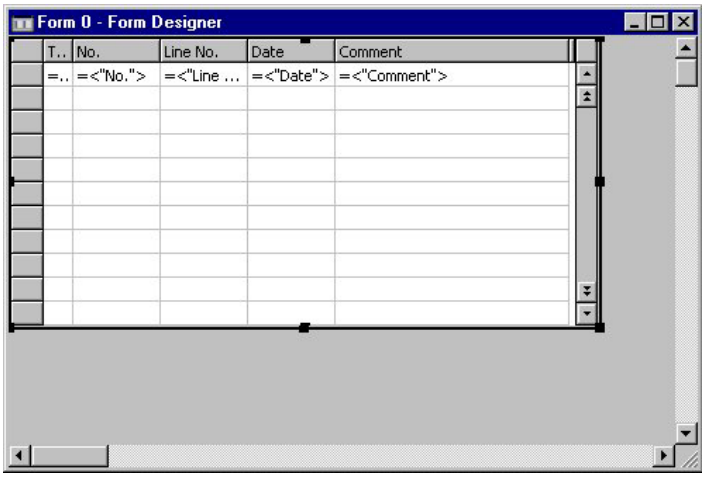
В этом упражнении вы создадите форму, которая отображает строки комментария, предназначенные для клиента. Это будет рабочая форма (worksheet form), которая помещается в сопутствующую форму на форме **Customer Comment**.

1 В окне Дизайнере объектов нажмите кнопку Form.

2 Нажмите кнопку New. Приложение C/SIDE отобразит окно **New Form**.

3 Создайте табличную форму, используя мастер, который содержит поля **Table Name**, **No.**, **Line No.**, **Date** и **Comment** из таблицы **Comment Line** (поле **Code** не требуется). См. упражнение 3.3. для получения дополнительной информации об использовании этого мастера.

4 Чтобы превратить эту форму в сопутствующую форму, вы должны удалить все кнопки (OK, Cancel и Help). Кроме того, необходимо переместить табличную зону в верхний левый угол и установить такой же размер этой формы, как у табличной зоны.



5 Закройте, сохраните и откомпилируйте форму с идентификатором *91501* и именем *Customer Comment Subform*.

6 Откройте форму.

7 Введите следующие записи:

Имя таблицы	**№**	**Строка №**	**Дата**	**Комментарий**
Customer	10000	10000	9/10/01	Cust. 10000 комментарий 1
Customer	10000	20000	9/11/01	Cust. 10000 комментарий 2
Customer	20000	10000	10/9/01	Cust. 20000 комментарий 1
Customer	30000	10000	10/10/01	Cust. 30000 комментарий 1

Какое свойство формы вы могли бы использовать, чтобы автоматически заполнить поле **Line No.**?

Какой цели служит поле **Table Name**?

Упражнение 3.6.3 Добавление сопутствующей формы в главную форму

В этом упражнении вы добавите сопутствующую форму в главную форму, установив элемент управления ”сопутствующая форма”.

1 Откройте форму 91500 в Дизайнере форм.

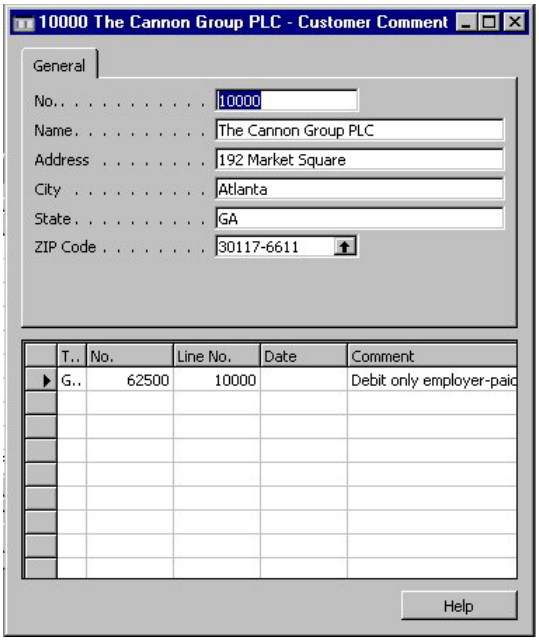
2 Нажмите элемент управления “сопутствующая форма”, чтобы выбрать его.

3 Откройте окно **Properties**.

4 Измените значение свойства SubFormID на *91501*.

5 Закройте, сохраните и откомпилируйте форму.

6 Откройте форму.



Что теперь появляется в том месте, где находится сопутствующая форма?

Полностью ли соответствует форма Customer Comment элементу управления “сопутствующая форма” (выровнена ли она по отношению к элементу управления “закладка”)?

Почему сопутствующая форма заключена в прямоугольник?

Что происходит теперь, когда вы изменяете размер формы, делая ее шире?

Что происходит теперь, когда вы изменяете размер формы, делая ее больше по высоте?

Какие записи появляются в сопутствующей форме, когда вы переходите к новому клиенту на главной форме?

Ниже приведен список того, что вы должны сделать, чтобы закончить построение этой комбинации форм.

1 Откорректируйте размеры.

2 Измените значение свойства VertGlue закладки на *Top*.

3 Измените значение свойства VertGlue и HorzGlue сопутствующей формы на *Both*

4 Откорректируйте вид формы в элементе управления “сопутствующая форма”.

5 Измените значение свойства Border сопутствующей формы на *None*.

6 Измените значения свойства Width и Height табличной зоны (на сопутствующей форме) на значения, которые вы записали ранее.

7 Отобразите записи в сопутствующей форме, которые соответствуют записи клиента в главной форме.

8 Измените значение свойства SubFormLink сопутствующей формы на *Table Name=CONST(Customer),No.=FIELD(No.)*. Вы должны установить Table Name в *Customer*, чтобы только комментарии клиента появлялись на форме.

9 Удалите ненужные поля из сопутствующей формы. Опасно показывать пользователю поля **Table Name**, **No.** и **Line No.** Форма автоматически заполняет эти значения, и пользователю нельзя дать возможность их изменять.

10 Измените значение свойства AutoSplitKey сопутствующей формы на *Yes*.

11 Удалите поля **Table Name**, **No.** и **Line No.** из сопутствующей формы.

12 Сохраните, откомпилируйте и снова протестируйте эти две формы.

Есть ли еще какие-нибудь проблемы с этими формами?

Можете ли вы вставить нового клиента?

Можете ли вы добавить комментарии к этому новому клиенту?

Что происходит, когда вы переходите к другому клиенту?

3.7 Типы и характеристики форм

В этом разделе описываются типы окон, которые пользователь видит в среде C/SIDE. Затем обсуждаются типы форм, которые вы будете создавать в своих приложениях, а также даются примеры и характеристики каждого из них.

Типы окон в среде C/SIDE

В среде C/SIDE 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.

Панели запросов (request panels)

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 картотечные 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.

Наш основной предмет

Формы, которые являются компоновочными блоками системы Navision, – это те объекты форм, которые связаны с таблицами (последними двумя из перечисленных выше). Хотя другие формы и окна важны, они не рассматриваются в этом разделе.

Почему это важно?

Обратите внимание, что хотя среда C/SIDE позволяет вам создавать множество типов форм с различным оформлением и методами функционирования, приложение Navision использует только некоторые из них. Это должно обеспечить единство оформления и стиля приложения.

Настоятельно рекомендуется в целях облегчения использования и единства стиля соблюдать эту политику при модификации системы Navision и любого приложения, которое взаимодействует с ней. В последующем описании полностью учтены стандарты системы Navision.

Типы форм и примеры

В системе Navision 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.

Понимание этих стандартов форм, используемых в системе Navision, даст вам возможность понять области применения, бросив беглый взгляд на эти объекты.

Тип	Одна запись/Несколько записей	Примеры
картотечная форма (Card)	Одна запись	карточка Клиент, карточка Поставщик, карточка Товар
статистическая форма (Statistics)	Одна запись	Статистика по клиентам, статистика по поставщикам, статистика по сотрудникам

Тип	Одна запись/Несколько записей	Примеры
табличная форма (Tabular)	Несколько записей	Валюта, условия оплаты
списочная форма (List)	Несколько записей	Список клиентов, список товаров, записи о товарах в гроссбухе
рабочая форма (Worksheet)	Несколько записей	Общая книга учета, книга учета денежных поступлений, книга учета перемещения товаров
колонтитульная форма (Header/Line)	Оба варианта (фактически две формы)	Счет-фактура, Posted Purchase Credit Memo, Finance Charge Memo
установочная форма (Setup)	Одна запись	Установка общего гроссбуха, информация о компании, установка продаж и дебиторов

Характеристики картотечной формы (Card)

Картотечная форма (Card) 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. Картотечные формы всегда содержат вкладки (например, индексные вкладки) у верхнего края, благодаря которым пользователь может выбирать и отображать различные группы полей.

Даже если полей немного, существует по крайней мере одна вкладка General, которая всегда стоит первой.

Поле первичного ключа Primary Key таблицы – всегда первое поле на вкладке General. В таблицах, где используются картотечные формы, имеется только одно поле в первичном ключе.

Присвоение имен картотечным формам

Картотечные формы получают имя таблицы, с которой они связаны, за ним следует слово «Card».

Например, картотечная форма, связанная с таблицей Customer, называется **Customer Card**. Картотечные формы также имеют по крайней мере одну кнопку меню в своей нижней части, получившую имя по имени той таблицы, что обеспечивает способы получения доступа к соответствующим данным.

Характеристики статистической формы (Statistics)

Статистическая форма (Statistics) – 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 статистическим формам

Статистические формы получают имя таблицы, с которой они связаны, за ним следует слово «Statistics».

Например, статистическая форма, связанная с таблицей **Customer**, называется **Customer Statistics**.

Статистическая формы ввода (Entry Statistics) – это специальная версия статистической формы. Они получает имя таблицы, с которой связаны, за ним следует словосочетание «Entry Statistics». Например, **Customer Entry Statistics**.

Характеристики табличной формы (Tabular)

Табличная форма (Tabular) – 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**, называется **Countries**.

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**.

Характеристики списочной формы (List)

Списочная форма (List) – 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 табличным 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» бухгалтерская форма (Ledger)

Более специализированная версия списочной формы – это бухгалтерская форма (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 в крайнем правом, а не крайнем левом столбце.

И, наконец, бухгалтерская форма получает имя связанной таблицы во множественном числе. Например, бухгалтерская форма, связанная с таблицей **Customer Ledger Entry**, называется **Customer Ledger Entries**.

Характеристики рабочей формы (Worksheet)

Рабочая форма (Worksheet) – 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 вместо этого остается в том же порядке, в каком пользователь ввел ее.

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».

Характеристики колонтитульной формы (Header/Line)

Многие формы в системе Navision 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 со вкладками, отображая один счет-фактуру. Однако строки счета отображаются в разделе формы, напоминающей таблицу, где несколько строк счета-фактуры (из того же счета-фактуры) можно просматривать одновременно и редактировать. Эти формы называются колонтитульными (Header/Line).

Колонтитульные формы являются фактически двумя отдельными формами, связанными с двумя различными таблицами. Главная форма – это картотечная 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 (подробности вы узнаете позже). Сопутствующая форма отвечает за взаимосвязь между этими двумя формами.

Присвоение имен колонтитульным формам

Во многих случаях колонтитульная форма будет представлять 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.

Эти случаи описаны более подробно в разделе, касающемся компонентных блоков Table.

Характеристики установочной формы (Setup)

Установочная форма (Setup) – 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. Причина в том, что там имеется много полей, эти формы содержат вкладки (например, индексные) у верхнего края для организации данных.

Поскольку там есть только одна запись, первичный ключ не отображается на этой форме.

Присвоение имен установочным формам

Установочные формы носят имя таблицы, с которой они связаны. Например, установочная форма, связанная с таблицей **General Ledger Setup**, называется **General Ledger Setup**.

Характеристики функциональной формы (Menu)

Функциональная форма (Menu) – несвязанная форма (не относящаяся

к какой-либо таблице), которая обеспечивает пользователю доступ ко многим другим типам форм для функциональной области из главного меню.

Эта форма обычно состоит из командных кнопок или кнопок меню. Однако внешний вид у этих кнопок бывает необычным. Свойства кнопок изменяются так, что они больше похожи на надписи с маленькими кругами или треугольниками перед заголовком. Тем не менее, эти кнопки функционируют так же, как обычные кнопки. Единственное различие – их вид.

Присвоение имен функциональным формам

Функциональные формы получают имя функциональной области, с которой они связаны, за ним следует слово «Menu». Например, функциональная форма, связанная с функциональной областью General Ledger, называется **General Ledger Menu**.

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 обеспечивают расширение функциональных возможностей системы Navision, но, подобно террасе дома, не влияют на основную архитектуру системы.

3.8 СТАНДАРТНЫЕ СРЕДСТВА НАВИГАЦИИ

Материал этого раздела покажет пользователю способы перехода от одной формы к другой. Мы описали механизм этого в предыдущем разделе, *«Расширение функциональных возможностей форм» (Extending the Functionality of Your Forms)*.

Здесь мы просто упомянем о стандартных средствах навигации, которые обеспечивают единство подходов во всей системе.

Картотечные и списочные

Большинство главных таблиц содержат как картотечную форму, так и списочную форму. Поскольку обе эти формы обращаются к одной и тоже таблице, существует много стандартов, определяющих действия каждой таблицы.

- Пользователь должен иметь возможность просматривать и редактировать по одной записи в картотечной форме.

- Пользователь должен иметь возможность просматривать (но не редактировать) все записи в списочной форме. Делая эту форму недоступной для редактирования, пользователь может начать поиск, вводя текст в определенном столбце.

- Пользователь должен иметь возможность вызывать списочную форму из картотечной формы следующими способами: клавиша F5, пункт меню List (на кнопке меню с тем же именем, что и у таблицы) и кнопка Lookup на панели инструментов. В этом случае пользователь должен иметь возможность выбирать новую главную запись и нажимать кнопку OK, чтобы выйти из списочной формы и заменить первоначальную картотечную форму на выбранную главную запись.

- Пользователь должен иметь возможность открывать новую картотечную форму из списочной формы (нажатием клавиш SHIFT+F5 или щелкнув пункт меню Card на кнопке меню с тем же именем, что у таблицы), которая отображает главную запись, выбранную в списочной форме.

Главные статистические формы

Как сказано выше, большинство главных таблиц содержат как картотечную форму, так и списочную форму. Кроме того, многие

главные таблицы содержат статистическую форму. Пользователь должен иметь возможность получать доступ к этой форме либо из картотечной 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 Statistics (на кнопке меню, у которой то же имя, что у таблицы).

Главная и бухгалтерская формы

Каждая главная таблица содержит, по меньшей мере, одну (обычно единственную) бухгалтерскую таблицу. Пользователь должен иметь возможность перейти из главных форм (картотечной, списочной и статистической) в бухгалтерскую форму. Он может сделать это тремя различными способами – используя drilldown-кнопку в поле FlowField, сочетание клавиш CTRL+F5 или пункт меню «Бухгалтерские записи» (Ledger Entries) (нажав на кнопку меню, имеющую то же имя, что у таблицы).

При нажатии drilldown-кнопки на экран выводятся только те записи, которые используются для вычисления значения (это функция, входящая во встроенные функциональные drilldown-возможности). Если используется сочетание клавиш или пункт меню, будут показаны все бухгалтерские записи 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

Для повышения производительности значение свойства RunFormView следует устанавливать во всех случаях, когда установлено значение свойства RunFormLink. Тот тип значения, который выбран в свойстве RunFormView, должен содержать поля, перечисленные в свойстве RunFormLink, иначе производительность снизится.

Учетные формы (Journal)

Каждая учетная форма (journal) содержит схожие кнопки в нижней части формы. Эти кнопки позволяют пользователю либо переходить к

другим формам, либо выполнять какие-либо действия.

Обычно на учетной форме расположены следующие кнопки.

- Кнопка меню, имеющая то же имя, что и главная таблица функциональной области. Эта кнопка меню должна содержать пункт меню s, позволяющий пользователю переходить прямо в картотечную форму для получения доступа к главной таблице или в бухгалтерскую форму для получения доступа к бухгалтерским записям. В обоих случаях выведенная форма должна содержать ссылку на запись главной таблицы, с которой связана данная строка книги учета.

- Кнопка меню с именем “Posting” («Регистрация»). Эта кнопка меню должна содержать следующие пункты меню, относящиеся к записи строк в книгу учета: «Запись» (Post), «Запись и печать» (Post and Print), «Отчет о тесте» (Test Report).

3.9 Ответы на вопросы

3.2.1 Создание пустой формы

Здесь нет вопросов.

3.2.2 Присоединение исходной таблицы

Каково значение в строке заголовка формы (обычно это область синего цвета наверху формы, содержащая кнопки Min, Max и Close)?

10000 The Cannon Group PLC – Пустая форма

Что происходит, когда вы щелкаете кнопку Next на панели инструментов?

Это приводит к изменению строки заголовка формы, где появляется следующий клиент из списка клиентов.

Форма считывает данные из исходной таблицы?

Да. По крайней мере, то, что необходимо для изменения строки заголовка.

3.2.3 Добавление простых элементов управления

Теперь форма содержит текстовую зону. Какая информация будет в ней отображаться?

Никакой.

Попытайтесь открыть форму, щелкнув File, Run. Что происходит?

Возникает ошибка. Элементы управления данными должны содержать выражение.

Связывание текстовых зон

Откройте форму и проверьте новую текстовую зону, перемещаясь в следующую, предыдущую, первую или последнюю записи.

Текстовая зона должна отобразить поле No. из таблицы клиентов.

Добавление надписи

Откройте форму и проверьте новую текстовую зону, перемещаясь в следующую, предыдущую, первую или последнюю записи. Надпись позволяет узнать, какое поле вы видите.

Текстовая зона должна отобразить поле No. из таблицы клиентов.

Добавление выключателя

Откройте форму и проверьте новый выключатель, перемещаясь в следующую, предыдущую, первую или последнюю записи. Кроме того, вам нужно поставить и убрать флажок.

В данный момент никакие пункты не заблокированы.

Вы можете изменить флажок Blocked.

3.2.4 Использование Add Label и Lock

Что еще было помещено на форму?

Надпись была присоединена к текстовой зоне.

Что происходит, когда вы перемещаете текстовую зону?

Надпись перемещается вместе с ней.

Что происходит, когда вы перемещаете надпись?

Перемещается сама надпись.

Как можно переместить только текстовую зону (подсказка: ищите курсор в форме руки)?

Выберите текстовую зону и переместите курсор к краю выбранного элемента, чтобы появился курсор в форме руки. Когда увидите курсор в форме руки, можете щелкнуть и перетащить текстовую зону.

Что происходит, если вы нажимаете кнопку Checkbox, а затем щелкаете форму?

Выключатель добавляется в форму вместе с надписью.

Использование Lock

Сколько текстовых зон вы добавили?

Четыре зоны.

Можете ли вы перемещать какие-либо элементы управления при выбранной кнопке Lock?

Нет. Это лишь приведет к созданию новой текстовой зоны.

Упражнение 3.3.1 Добавление контейнерных элементов управления

Использование закладки

Что произошло с элементом управления “закладка”?

Было добавлено три вкладки – Test, Frame и Table Box.

Добавление элементов управления на вкладку

Что происходит, если вы нажимаете на вкладку General и затем снова на вкладку Test?

Элемент управления исчезает, когда выбрана вкладка General, и снова появляется, когда выбрана вкладка Test.

Отредактируйте свойство PageNames закладки так, чтобы он выглядел так: General,Frame,Table Box. Где теперь находится надпись?

На вкладке Frame (вторая вкладка в элементе управления “закладка”).

Переместите элемент управления ”надпись” в нижнюю часть формы (в сторону от элемента управления “закладка”). Что происходит, когда вы переходите на вкладку General и снова возвращаетесь назад?

Надпись все время видна.

Нажмите вкладку General и переместите надпись из основания формы в середину страницы для вкладки General. Что происходит, когда вы переходите на вкладку Frame и снова возвращаетесь назад?

Элемент управления исчезает, когда выбрана вкладка Frame, и

снова появляется, когда выбрана вкладка General. Она снова находится в элементе управления “закладка”.

Добавление фрейма

Что происходит, когда вы перемещаете фрейм?

Все элементы управления во фрейме перемещаются вместе с ним.

Что происходит, когда вы удаляете фрейм?

Все элементы управления во фрейме тоже удаляются.

Добавление табличной зоны

Откройте форму. Что отображается в табличной зоне?

Несколько строк записей из таблицы товаров. В прямоугольнике серого цвета рядом с текущей строкой находится треугольник.

Что происходит со строкой заголовка формы, если вы щелкаете в другой строке табличной зоны?

Она превращается в строку, выбранную в табличной зоне.

Что происходит, если вы разворачиваете форму или просто увеличиваете ее?

Размеры и расположение элементов управления не изменяются.

Масштабирование элементов управления

Откройте форму. Что происходит теперь, когда вы изменяете размер формы, делая ее больше?

Закладка и табличная зона увеличиваются вместе с формой.

3.3.2 Отображение опций

Использование текстовой зоны

Какая кнопка находится в текстовой зоне?

Кнопка DropDown.

Что происходит, когда вы нажимаете кнопку?

Краткий список возможностей отображается под текстовой зоной.

Использование кнопок выбора

Сколько кнопок выбора отобрано?

Одна.

Что происходит, если вы пытаетесь отобрать еще одну?

Отменен выбор первой кнопки выбора.

Что происходит, если вы изменяете текстовую зону, которая связана с полем Costing Method, на Specific?

Выбрана третья кнопка, обозначенная как Specific.

Что происходит, если вы изменяете текстовую зону, которая связана с полем Costing Method, на Average?

Ни одна из кнопок не выбрана, потому что кнопки для Average нет.

3.3.3 Отображение индикаторов выполнения

Используйте кнопку Next, чтобы просмотреть несколько различных записей. Какой ряд значений вы увидели?

Все – от 15% до 70%.

Что происходит, если вы изменяете значение свойства MaxValue на *1000* и снова открываете форму?

Значения на индикаторе едва видны. Процент становится менее 1.

3.3.4 Отображение BLOB-полей (рисунков)

Используйте кнопку Next, чтобы просмотреть несколько различных записей. Перечислите некоторые элементы, содержащие изображения.

70102 – Краска, синий

1896-S – Афинское отделение

1906-S – Афинское мобильное основание

3.4.1 Картотечные формы

Запуск мастера

Здесь нет вопросов.

Мастер картотечной формы

Здесь нет вопросов.

3.4.2 Табличные формы

Мастер табличной формы

Что происходит?

Окно Find появляется и начинает поиск того, что я печатаю.

Почему это желательно пользователю?

Пользователь не должен запоминать, как открывать окно Find. Можно просто щелкнуть столбец и начать печатать. Это быстрее.

3.4.3 Использование Field Menu

Добавление полей в картотечную форму

Работает ли поле должным образом?

Да.

Что еще помещено на форму вместе с текстовой зоной?

Надпись, присоединенная к текстовой зоне.

Добавление полей на табличную форму

Работает ли поле должным образом?

Да.

Каков был заголовок столбца?

Баланс. Имя поля.

Считаете ли вы, что большинству пользователей необходимо видеть поле Balance на этой форме?

Нет. Только некоторые пользователи будут использовать это поле в форме.

3.4.4 Как сделать столбцы невидимыми на табличной форме

Видите ли вы столбец?

Нет.

Щелкните View, Show Column. Находится ли поле Balance в списке? Находится ли поле Net Change в списке?

Да. Поле Balance – в списке.

Нет. Поле Net Change – не в списке.

Почему поле Balance внесено в список, а поле Net Change - нет?

Поле Net Change не добавлено на форму. Только добавленные на форму поля отображаются здесь.

Поставьте флажок рядом с полем Balance и нажмите кнопку OK. Отображается ли теперь поле Balance?

Да.

Что происходит, если вы закрываете форму и открываете ее снова?

Поле Balance все еще находится на форме, поэтому пользователь может видеть его.

3.5.1 Создание командных кнопок

Какой заголовок имеет кнопка?

OK.

Что происходит, когда вы нажимаете кнопку?

Форма закрывается.

Снова откройте форму. Медленно измените размеры формы. Что происходит с вашей новой кнопкой?

Она остается на месте и не остается рядом с кнопкой Help.

Что произошло с кнопкой Help?

Она перемещается. Она остается в нижнем правом углу формы.

Что заставило их действовать по-разному?

Связующие свойства.

Снова протестируйте форму, изменив ее размеры. Что происходит с вашей новой кнопкой?

Она остается около кнопки Help в нижнем правом углу формы.

3.5.2 RunObject PushAction

Где находится кнопка? Где находятся кнопки OK и Cancel?

Она там, где была поставлена, но кнопки OK и Cancel исчезли.

Они не видны.

Что происходит, когда вы нажимаете кнопку?

Ничего.

Что вы не установили?

Свойство PushAction.

Снова откройте форму. Что происходит на этот раз при нажатии кнопки Card?

Созданная мной картотечная форма открыта.

Закройте картотечную форму. Выберите клиента 50000 в форме списка. Нажмите кнопку Card. Какой клиент отображен на картотечной форме?

Первый клиент 10000.

3.5.3 Формы для связывания

Что происходит, когда вы нажимаете кнопку Card при условии, что выбран клиент 50000?

На картотечной форме отображается клиент 50000.

Что происходит, если вы закрываете картотечную форму, выбираете другого клиента и снова нажимаете кнопку Card ?

Отображается этот клиент (выбранный в списочной форме).

Обратите внимание, что пока картотечная форма открыта, вы можете снова щелкнуть списочную форму и выбрать другую запись. Что происходит с картотечной формой, когда вы изменяете списочную форму таким образом?

Ничего. По-прежнему отображается запись другого клиента.

Сконструируйте форму и измените значение свойства RunFormLinkType кнопки Card на *OnUpdate*. Сохраните изменения и откройте форму. Что теперь происходит, когда вы изменяете форму списка, пока картотечная форма открыта?

Она тоже изменяется, переходя к новому клиенту. Она остается связанной при каждом обновлении списочной формы.

3.5.4 Создание кнопок меню

Что происходит, когда вы нажимаете новую кнопку меню?

Ничего.

3.5.5 Создание пунктов меню

Что теперь происходит, когда вы нажимаете кнопку меню?

Меню из добавленных мной кнопок меню отображается под кнопкой меню.

Что происходит, когда вы нажимаете пункт меню Card?

Картотечная форма отображается для выбранной записи.

Что происходит, когда вы нажимаете сочетание клавиш Shift+F5?

Открывается картотечная форма с выбранной записью. То же, что после щелчка на пункте меню.

3.5.6 LookupTable PushAction

Какая форма появляется теперь, когда вы нажимаете функциональную клавишу F5?

Форма 22, обычный список клиентов Customer List.

Почему появляется эта форма, а не ваша форма списка клиентов?

Форма поиска по умолчанию для таблицы клиента – это форма Customer List. Моя форма Customer List не установлена как форма поиска по умолчанию. Свойство LookupformID таблицы клиента инструктирует систему, какую форму вывести.

Сконструируйте таблицу 18 и измените значение свойства LookupFormID таблицы на *91301*. Снова откройте свою картотечную форму. Какая форма появляется теперь, когда вы нажимаете функциональную клавишу F5?

Форма 91301, My Customer List.

3.6.1 Создание главной формы

Откройте окно свойств для отображения свойств сопутствующей формы и запишите свойства Height и Width (“высота и ширина”).

Height *5500*

Width *9350*

Что появляется там, где находится сопутствующая форма?

Ничего.

3.6.2 Создание сопутствующей формы

Какое свойство формы вы могли бы использовать, чтобы автоматически заполнить поле **Line No.**?

AutoSplitKey

Какой цели служит поле **Table Name**?

Эта таблица может хранить комментарии из нескольких главных таблиц. Table Name изменяет отношение таблиц поля No. на соответствующую главную таблицу.

3.6.3 Добавление сопутствующей формы на главную форму

Что теперь появляется в том месте, где находится сопутствующая форма?

Форма 91501

Полностью ли соответствует сопутствующая форма Customer Comment Subform элементу управления “сопутствующая форма” (выровнена ли она по отношению к элементу управления “закладка”)?

Не совсем. Ширина правильная. Полосы прокрутки справа и внизу – не на обычном месте.

Почему сопутствующая форма заключена в прямоугольник?

Это граница элемента управления “сопутствующая форма”.

Что происходит теперь, когда вы изменяете размер формы, делая ее шире?

Сопутствующая форма не увеличивается.

Что происходит теперь, когда вы изменяете размер формы, делая ее больше по высоте?

Закладка увеличивается и переходит под сопутствующую форму. Сопутствующая форма по-прежнему не увеличивается.

Какие записи появляются в сопутствующей форме, когда вы переходите к новому клиенту на главной форме?

Все записи всех клиентов.

Есть ли еще какие-нибудь проблемы с этими формами?

Нет.

Можете ли вы вставить нового клиента?

Да.

Можете ли вы добавить комментарии к этому новому клиенту?

Да.

Что происходит, когда вы переходите к другому клиенту?

Сопутствующая форма изменяется так, что я вижу только комментарии для нового клиента.

3.10 Обзор главы

В этом разделе содержатся вопросы, касающиеся материала данной главы. Ответы на вопросы можно найти в подразделе «Ответы».

Вопросы

Верно или неверно следующее утверждение? Все формы являются связанными.

Верно или неверно следующее утверждение? Мастер форм не используется.

Какое свойство необходимо установить на кнопке Command, чтобы свойство RunObject работало?

Какой элемент управления позволяет отображать на форме более одной записи одновременно?

Какой элемент управления позволяет отображать на форме другую форму?

Какое свойство текстовой зоны связывает ее с полем в таблице?

Какое свойство формы связывает форму с таблицей?

Как превратить пункт меню в подменю на кнопке меню?

Как присоединить надпись к другому элементу управления так, чтобы надпись при перемещении перемещалась вместе с другим элементом управления?

Какие действия необходимо выполнить для создания списочной формы из таблицы?

Ответы

Все формы являются связанными.

Неверно. Некоторые формы не привязаны ни к какой таблице (не связаны). Например, форма Main Menu.

Мастер форм не используется.

Неверно. Большинство форм можно создать с помощью мастера. Обычно все, что требуется добавить, – это кнопки.

Какое свойство необходимо установить на кнопке Command, чтобы свойство RunObject работало?

Вы должны установить PushAction на RunObject

Какой элемент управления позволяет отображать на форме более одной записи одновременно?

Tablebox

Какой элемент управления позволяет отображать на форме другую форму?

Subform

Какое свойство текстовой зоны связывает ее с полем в таблице?

SourceExpr

Какое свойство формы связывает форму с таблицей?

SourceTable

Как превратить пункт меню в подменю на кнопке меню?

Сделайте отступ для других пунктов меню под ним, используя кнопку Right в окне Menu Designer.

Как присоединить надпись к другому элементу управления так, чтобы надпись при перемещении перемещалась вместе с другим элементом управления?

Щелкните кнопку Add Label в окне Toolbox перед добавлением

другого элемента управления.

Какие действия необходимо выполнить для создания списочной формы из таблицы?

Создайте табличную форму для таблицы с использованием мастера.

Измените значение свойства Editable формы на False.

Добавьте соответствующие кнопки для перехода в Card, Ledger Entries и Statistics.

Глава 4

Основные отчеты

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.

Глава содержит следующие разделы.

- Отчеты;
- Дизайнеры отчетов;
- Как работают разделы;
- Некоторые полезные свойства;
- Группировка в отчете;
- Суммирование в отчете;
- Добавление расширенных возможностей;
- Типы отчетов;
- Создание основного отчета;
- Ответы на вопросы;
- Обзор главы.

4.1 Отчеты

Отчеты в среде C/SIDE служат нескольких целям.

- Отчеты используются для вывода в структурированном виде на печать информации из базы данных. Например, в приложении заказа на покупку вы можете создать отчет, содержащий список всех клиентов, где перечислены все заказы, сделанные каждым клиентом.
- Все создаваемые документы, относящиеся к приложению, должны быть сделаны в виде отчета. Например, чтобы напечатать счет-фактуру, вы создадите 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.

На следующей диаграмме представлены компоненты отчета и их взаимоотношение.

Компоненты отчета

Из приведенной диаграммы видно, что отчет составлен из ряда различных компонентов. Ниже дано краткое описание каждого компонента.

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)*, предназначенные для вывода на печать результата вычисления любого допустимого C/AL-выражения, такого как содержание поля таблицы (а также сложных вычислений), *надписи (labels)* для вывода на печать статического текста, такого как заголовок столбца данных, а также *формы(shapes)*, *изображения (images)* и *зоны изображения (picture boxes),* предназначенные 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.

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 листе Property Sheet 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. Триггеры модифицируются в Редакторе C/AL-кода.

Логическое и визуальное конструирование

В конструировании отчета есть две стороны: определение логической структуры, модели данных и разработка визуальной компоновки.

Определение модели данных – это определение того, как собираются данные для отчета. Сюда входит следующее:

- определение таблиц, которые будут использоваться отчетом для создания элементов данных;
- 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, какие элементы пользователь сможет изменять во время выполнения;
- 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

Модель данных отчета строится на основе элементов данных. Элемент данных относится к таблице. При выполнении отчета (см. диаграмму на стр. 214), каждый элемент данных повторяется для всех записей в соответствующей таблице. Когда отчет основан на нескольких таблицах, вы устанавливаете иерархию элементов данных для контроля над тем, как информация собирается, задавая 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 – тот, который соответствует таблице **Customer,** и тот, который соответствует таблице **Sales Order**. Второй элемент данных выделен отступом (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

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 напечатана (например, заголовки столбца), но существуют отчеты – например, примеры есть в лабораторном руководстве (lab manual) – где раздел области данных не используется вообще, и вся информация печатается в других разделах.

На следующем рисунке показан законченный отчет.

Разделы заголовков

Sales Statistics
CRONUS International Inc.
Customer: No.: 10000..30000

09/28/95 10/28/95 11/28/95

Разделы области данных

Разделы нижнего колонтитула

No.	Name	...before	10/27/95	11/27/95	12/27/95	after...
10000	**Kontorforsyningen A/S**					
	Sales (LCY)	57,509.00	0.00	0.00	0.00	0.00
	Profit (LCY)	12,655.00	0.00	0.00	0.00	0.00
	Profit %	22.0	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
20000	**Ravel Møbler**					
	Sales (LCY)	1,525.00	0.00	0.00	0.00	0.00
	Profit (LCY)	335.00	0.00	0.00	0.00	0.00
	Profit %	17.9	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
30000	**Lauritzen Kontormøbler A/S**					
	Sales (LCY)	13,676.20	0.00	0.00	0.00	0.00
	Profit (LCY)	2,444.20	0.00	0.00	0.00	0.00
	Profit %	17.9	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
Total						
	Sales (LCY)	72,710.20	0.00	0.00	0.00	0.00
	Profit (LCY)	15,434.20	0.00	0.00	0.00	0.00
	Profit %	21.2	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

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 (title) и информацию о фильтре, который использовался по отношению к номерам клиентов.
- Каждый раздел области данных содержит информацию о клиенте, размещенную на нескольких строках. Строки «Profit %» 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.

4.2 Дизайнеры отчетов

Дизайнер отчетов (Report Designer) содержит два дополнительных дизайнера – Дизайнер разделов (Section Designer), используемый для компоновки отчетов, и Дизайнер форм вариантов запроса (Request Options Form Designer), используемый для конструирования форм вариантов запроса.

Дизайнер отчетов

Выбирая отчет в Дизайнере объектов (Object Designer) и нажимая кнопку Design, вы переходите в Дизайнер отчетов.

В окне **Report Designer** вы определяете модель данных, добавляя элементы данных.

В окне **Report Designer** вы можете определить таблицы, которые будут считываться при запуске отчета. Важно обратить внимание на то, что любая упомянутая здесь таблица будет считана полностью при выполнении отчета. Чтобы ограничить число считываемых записей, вы должны связать и профильтровать элементы данных. Фактически вы могли бы считать одну и ту же таблицу несколько раз в одном отчете, внеся его несколько раз в список в Дизайнере отчетов. Например, вам может потребоваться считывать сначала данные незаблокированных, а затем – заблокированных клиентов.

Таблицы, которые вы вносите в список, становятся элементами данных в отчете. Элементы данных – это переменные записей (как переменные в коде), которые позволяют отчету получать доступ к данным таблицы.

Правый столбец в Дизайнере отчетов– это Name, которое будет использовать элемент данных во всем отчете. Подобно всем именам переменных, эти имена должны быть уникальными в пределах отчета.

Вы можете использовать Дизайнер отчетов для определения того, как механизм отчета должен обеспечивать считывание соответствующих таблиц.

Упорядочение элементов данных

Порядок, в котором перечислены элементы данных, является порядком считывания данных из таблиц отчетом. В нижней части окна **Report Designer** имеется четыре кнопки со стрелкой. Вы можете использовать кнопки со стрелкой «вверх» и «вниз», чтобы изменить порядок элементов данных.

Установка отступа для элемента данных

Вы можете использовать правую и левую кнопки для установки или удаления отступа для элемента данных. Установление отступа для одного элемента данных под другим элементом данных влияет на отчет следующим образом:

- все записи, считываемые начиная с верхнего элемента данных, будут считываться от элемента данных с отступом (при этом учитываются все фильтры и связи). (Это напоминает вложенный цикл FOR).

Связывание элемента данных

Каждый элемент данных имеет свойство с именем DataItemLinkReference и свойство с именем DataItemLink. Обычно они используются для определения отношения Master/Detail между этими двумя таблицами. В свойстве LinkReference вы можете выбрать, с каким другим элементов данных вы хотели бы установить связь (это должен быть элемент данных, под которым расположен данный элемент данных с отступом). В свойстве DataItemLink вы можете определить фильтр для этого элемента данных, подобрав в данном элементе данных поля, соответствующие полям из элемента данных LinkReference.

Примечание

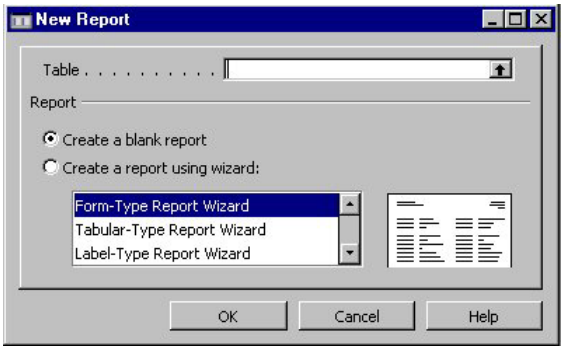
Для элемента данных с отступом необходимо выбрать подходящий ключ. Он должен содержать поля, которые используются для связывания элемента данных.

Упражнение 4.2.1 Создание модели данных

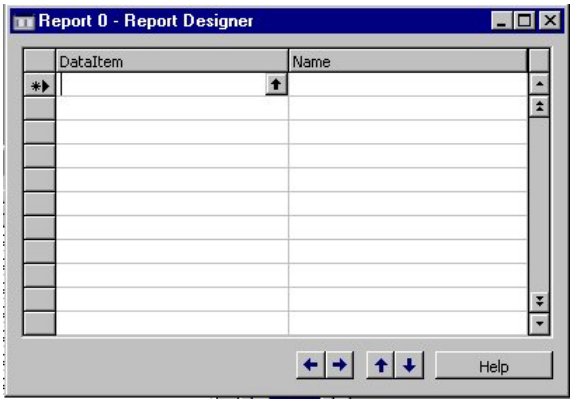
Это упражнение поможет вам понять некоторых основные принципы конструирования отчетов в Navision. В этой части упражнения вы создадите модель данных в Дизайнере отчетов.

1 Откройте Дизайнер объектов и щелкните Report.

2 Нажмите кнопку New. Появится окно **New Report**. Вы не будете использовать мастер отчетов, потому что он ограничивает вас работой с одной таблицей.



3 Нажмите кнопку OK. Появится окно Дизайнера отчетов.

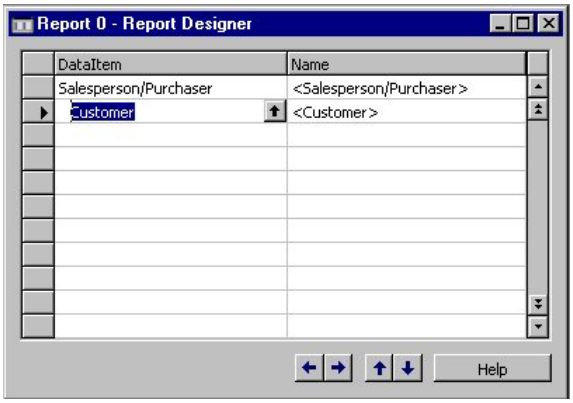


4 Выберите первую пустую строку в Дизайнере отчетов и введите номер или имя таблицы **Salesperson/Purchaser** (13).

5 Выберите первую пустую строку в Дизайнере отчетов и введите номер или имя таблицы **Customer** (18).

Поскольку вы хотите, чтобы отчет искал клиентов по продавцу, вы должны установить его так, чтобы он сначала считывал таблицу **Salesperson/Purchaser**. Затем для каждой записи продавца он будет считывать всех клиентов для этого продавца. Это делается посредством смещения элемента данных клиента под элементом данных продавца и их связывания. Подробные инструкции даны ниже.

1 Выделите отступом элемент данных Customer, щелкнув правую кнопку со стрелкой в нижней части формы.



При выбранном элементе данных клиента получите свойства этого элемента данных.

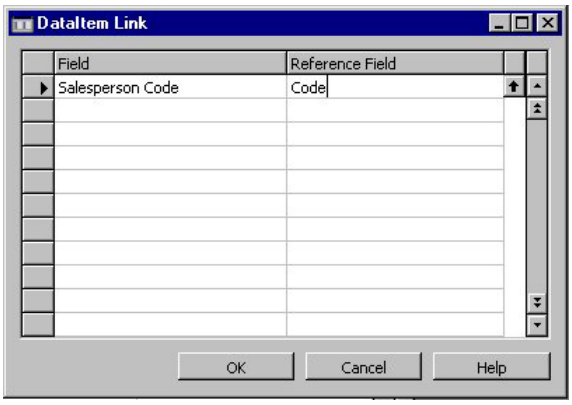
2 Щелкните свойство DataItemLinkReference. Убедитесь в том, что оно установлено на *Salesperson/Purchaser*.

3 Щелкните свойство DataItemLink. Вы должны подобрать поля, которые связывают эти две таблицы.

4 Нажмите кнопку поясняющего редактирования (assist-edit).

5 В столбце Field выберите поле **Salesperson Code** из таблицы **Customer**.

6 В столбце Reference выберите поле **Code** из таблицы **Salesperson/Purchaser**.



7 Нажмите кнопку OK, чтобы сохранить связь.

При связывании двух элементов данных вы должны всегда сортировать выделенный отступом элемент данных соответственно. В этом случае необходимо изменить свойство DataItemTableView элемента данных Customer на *SORTING(Salesperson Code)*. Закройте, сохраните и откомпилируйте отчет с идентификатором 91200 и именем *Customers by Salesperson 1*.

Модель данных теперь закончена. Следующий шаг – проектирование страницы. Прежде чем переходить к нему, вы должны узнать о Дизайнере разделов.

Дизайнер разделов

Как вы помните, каждый элемент данных может содержать один или несколько разделов, определенных в отчете. Дизайнер разделов позволяет конструировать визуальную компоновку отчета раздел за разделом. Находясь в Дизайнере отчетов, выберите меню View, а затем Sections.

Примечание

Если свойство Processing Only отчета установлено в *Yes,* вы не можете создавать какие-либо разделы. Это свойство обсуждается позже наряду с другими свойствами.

Каждый элемент данных может содержать любое число разделов (помните, что каждый раздел – подобъект одного из элемента данных). Поэтому при создании раздела (F3) вы должны выбрать элемент данных, с которым связан раздел, и определить тип необходимого раздела. Вы можете удалить раздел, щелкнув серый прямоугольник и затем нажав клавишу F4.

В следующей таблице дано краткое описание типов разделов в отчете.

Раздел	Описание
Область данных (Body)	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.
Нижний	Этот раздел печатается последним для элемента

Раздел	Описание
колонтитул (Footer)	данных. По умолчанию он печатается только один раз при каждой обработке элемента данных, но вы можете выбрать возможность печатать его на каждой странице. Вы можете также определить, чтобы этот раздел печатался в нижней части страницы, что гарантирует, что раздел будет помещен на как можно ближе к нижней границе страницы.
Групповой нижний колонтитул (GroupFooter)	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.
Групповой заголовок (GroupHeader)	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.
Заголовок (Header)	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.
Промежуточный нижний колонтитул (TransFooter)	Этот раздел печатается в нижней части страницы для элемента данных, вывод на печать которого будет продолжено на следующей странице.
Промежуточный заголовок (TransHeader)	Этот раздел печатается в верхней части страницы для элемента данных, вывод на печать которого продолжается с предыдущей страницы.

Каждый раздел функционирует подобно маленькой форме со свойством SourceTable, значением которого является таблица элемента данных. Элементы управления в разделах не содержат триггеры, но имеют те же свойства, что и на форме.

Примечание

Хотя вы видите цвета, которые устанавливаете при конструировании раздела, эти цвета НЕ будут напечатаны, когда будет выполняться отчет. Единственный способ вывода на печать цвета заключается в том, чтобы использовать элемент управления Image или PictureBox.

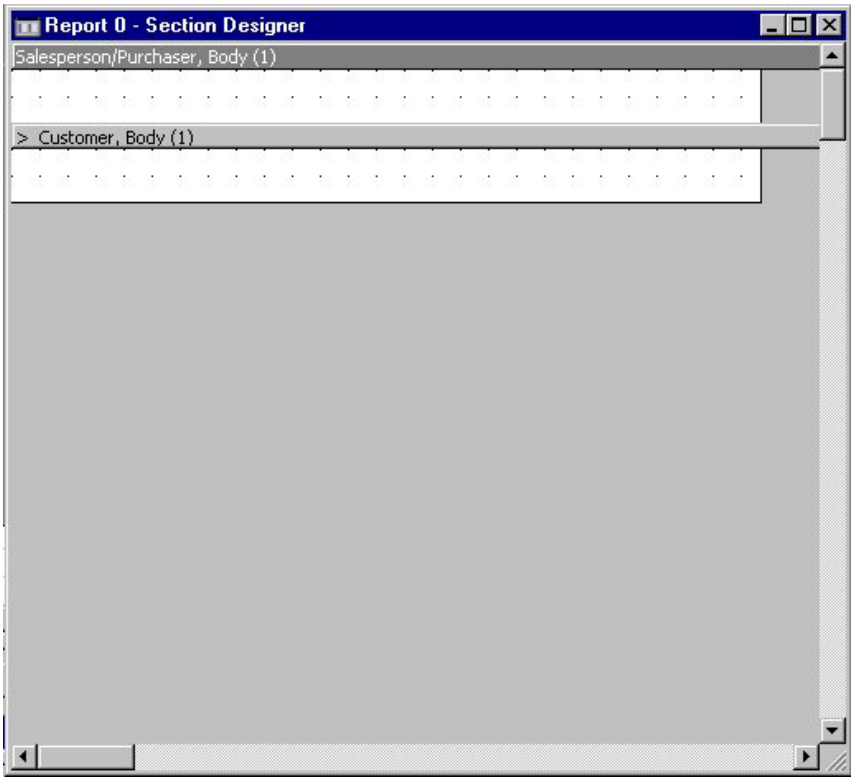
Упражнение 4.2.2 Формирование композиции страницы

В этом упражнении вы определите разделы и закончите составление отчета, начатое ранее.

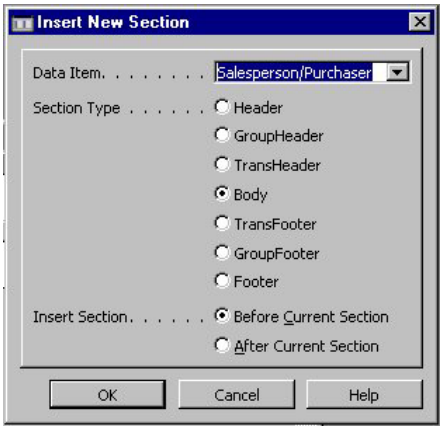
1 Откройте отчет 91200 в Report Designer.

2 В главном меню нажмите View, Sections. Вы должны увидеть два раздела, которые уже были созданы отчетом:

- область данных Salesperson/Purchaser (1)
- область данных Customer (1).



3 Нажмите клавишу F3, чтобы вставить другой раздел. Появится окно **Insert Section**.



4 Выберите элемент данных Salesperson/Purchaser и тип раздела Header. Это приведет к вставке заголовка Salesperson в разделы (раздел заголовка для элемента данных будет всегда стоять перед разделом области данных).

5 Нажмите кнопку OK, чтобы вставить нужный раздел.

Обратите внимание на появление нового прямоугольника серого цвета с именем Salesperson/Purchaser Header(1).

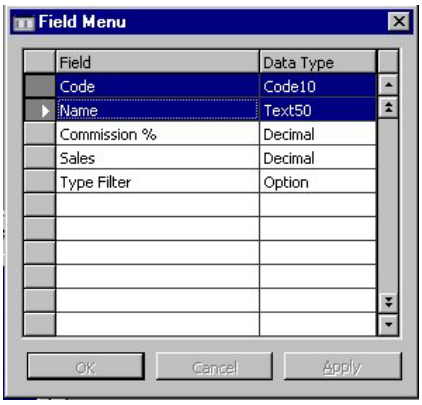
6 Повторите шаги 4 и 5, чтобы вставить раздел заголовка Customer.

Теперь в вашем отчете должно быть четыре раздела. Но в них нет никаких элементов управления. Если бы вы выполнили отчет, на страницах ничего не было бы напечатано .

7 Нажмите раздел области данных Salesperson/Purchaser.

8 Нажмите View, Field Menu. Появится окно **Field Menu**.

9 Нажмите клавишу CTRL и выберите поля **Code** и **Name**.



10 Поместите курсор в раздел области данных Salesperson/Purchaser.

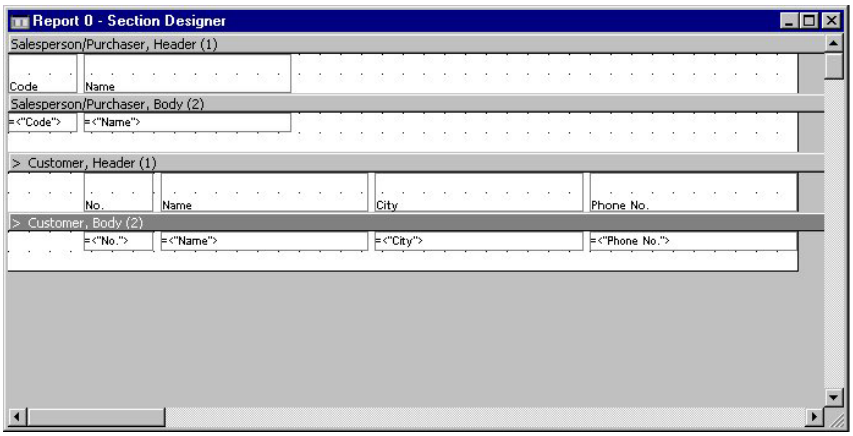
11 Нажмите один раз, чтобы активировать Дизайнер разделов.

12 Нажмите снова, чтобы создать поля в разделе.

Обратите внимание, что созданные в разделе отчета надписи и текстовые зоны не такие, как на форме. Надписи помещены над текстовыми зонами в столбцах. Вероятно, ваши надписи находятся на текстовых зонах, потому что раздел был слишком мал, чтобы поместить их над зонами. Вы можете передвигать элементы управления так же, как вы это делали с формами.

13 Перетащите надписи поодиночке от раздела области данных в раздел заголовка и выровняйте их по отношению к соответствующим текстовым зонам.

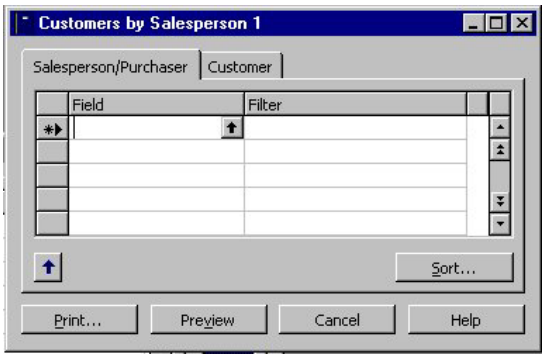
14 Повторите шаги 7-13 для конструирования раздела области данных Customer. Добавьте поля **No.**, **Name**, **City** и **Phone No.** из таблицы **Customer**. Не забудьте переместить надписи в раздел заголовка.



15 Закройте, сохраните и откомпилируйте отчет. Вам не обязательно сохранять отчет перед тем, как выполнять его, но это очень желательно. Никогда нельзя быть уверенным в стабильной работе компьютера. Регулярно сохраняйте работу при конструировании отчетов!

16 Откройте отчет. Появится форма запроса. Эта форма позволяет пользователю вводить фильтры для элементов данных или изменять сортировку элементов данных. Можно также выбрать

возможность вывода отчета на печать или его предварительного просмотра.



17 Нажмите кнопку Preview.

Примечание

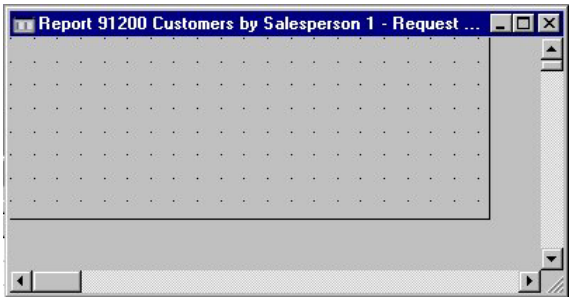
Если драйвер принтера для Windows не загружен, вы не сможете печатать или предварительно просматривать какие-либо отчеты.

Отчет выглядит так, как вы ожидали? Где печатаются разделы заголовка для каждого элемента данных?

Повторялись ли разделы заголовка?

Дизайнер форм вариантов запроса

Находясь в Дизайнере отчетов, вы можете выбрать меню View, а затем форму запросов (Request Form).



Используйте Дизайнер форм вариантов запроса (Request Options Form Designer) для конструирования вкладки **Options** формы запроса. Форма запроса – это форма, которая отображается для пользователя, когда он выполняет отчет.

Вы не можете отсюда конструировать вкладки элементов данных или кнопки в нижней части формы запроса. Это относится к любой форме, которую вы создаете. Помните, что вкладка **Options** будет последней вкладкой на форме запроса.

Когда вы добавляете элементы управления к этой форме, вкладка **Options** будет отображена. Вкладка **Options** не будет видна пользователю, пока вы не добавите некоторые элементы управления.

Обычно необходимо включать некоторые триггеры перед тем, как использовать вкладку **Options** формы запроса. Поскольку данный курс охватывает только основные отчеты, вам не понадобится добавлять элементы управления к форме запроса.

Другие вкладки формы запроса можно изменять через свойства, а как – вы увидите в остальной части этой главы.

4.3 Как работают разделы

Здесь вы изучите, как определенные типы разделов могут использоваться и когда они выводятся на печать.

Упражнение 4.3.1 Заголовки, области данных и нижние колонтитулы

В этом упражнении вы поэкспериментируете с разделами заголовка, области данных и нижнего колонтитула. Сначала без указания отступа и связывания, затем с отступом и, наконец, со связыванием.

Без отступа и связывания

1 Откройте Дизайнер объектов и щелкните Report.

2 Нажмите кнопку New, и появится окно **New Report**. Вы не будете использовать мастер отчетов, потому что он ограничивает вас работой с одной таблицей.

3 Нажмите кнопку OK, и появится Дизайнер отчетов.

4 Щелкните первую пустую строку в Дизайнере отчетов и введите номер или имя таблицы **Salesperson/Purchaser** (13).

5 Щелкните первую пустую строку в Дизайнере отчетов и введите номер или имя таблицы **Customer** (18). Не выделяйте отступом и не связывайте два этих элемента данных.

6 Щелкните элемент данных Customer.

7 Откройте окно **Properties**.

8 Измените значение свойства MaxIteration на *5*.

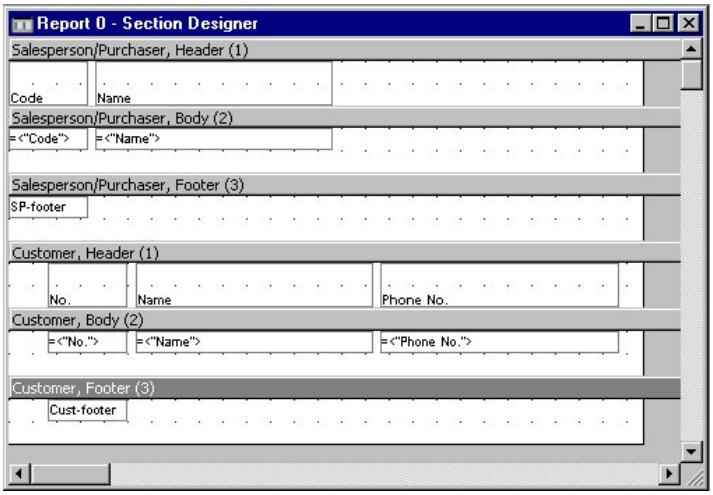
9 Нажмите кнопку View, Sections.

10 Вставьте заголовок для каждого элемента данных. Теперь у вас должно быть четыре раздела. Вставьте также нижний колонтитул для каждого элемента данных.

11 В Salesperson Body разместите поля **Code** и **Name** и перетащите надписи в текстовую зону.

12 В Customer Body поместите поля **No.**, **Name** и **Phone No.** и переместите надписи в заголовок.

13 В Footer для каждого элемента данных поместите надпись из панели инструментов и измените заголовок на *SP-footer* и *Cust-Footer* соответственно.



14 Закройте, сохраните и откомпилируйте отчет с идентификатором *91300* и именем *Basic Sections Test*.

15 Откройте отчет.

Как отчет прошел цикл через таблицы?

Где печатается раздел Header каждого элемента данных?

Где печатается раздел Footer каждого элемента данных?

С отступом

1 Откройте отчет в Дизайнере отчетов.

2 Выделите отступом элемент данных Customer (пока не связывайте его).

3 Закройте, сохраните и откомпилируйте отчет.

4 Откройте отчет.

Как отчет прошел цикл через таблицы?

Где печатается раздел Header каждого элемента данных?

Где печатается раздел Footer каждого элемента данных?

Со связыванием

1 Откройте отчет в Дизайнере отчетов.

2 Выберите элемент данных Customer.

3 Измените значение свойства DataItemLink на *Salesperson Code=FIELD(Code)*.

4 Помните, что необходимо также установить сортировку для отчета. Добавьте SalesPerson Code Key в таблицу **Customer**. Измените значение свойства DataItemTableView на *SORTING(Salesperson Code)*.

5 Закройте, сохраните и откомпилируйте отчет.

6 Откройте отчет.

Как отчет прошел цикл через таблицы?

Где печатается раздел Header каждого элемента данных?

Где печатается раздел Footer каждого элемента данных?

Упражнение 4.3.2 TransHeaders и TransFooters

В этом упражнении вы добавите промежуточные заголовки TransHeader и TransFooter к предыдущему отчету.

1 Откройте отчет 91300 в Дизайнере отчетов.

2 Выберите элемент данных Customer.

3 Откройте окно **Properties**.

4 Удалите значение в свойстве MaxIteration.

5 Закройте, сохраните и откомпилируйте отчет.

6 Откройте отчет.

Обратите внимание, что Customer, предназначенные для некоторых продавцов, занимают более одной страницы. Однако на второй

странице отчет не печатает какие-либо заголовков, которые сообщали бы нам, какого продавца мы видим. Был бы хорошо, если бы все печатаемые страницы были с заголовками.

1 Откройте отчет 91300 в Дизайнере отчетов.

2 Нажмите кнопку View, Sections.

3 Выберите каждый из разделов Header в отчете и измените значение свойства PrintOnEveryPage на *Yes*.

4 Закройте, сохраните и откомпилируйте отчет.

5 Откройте отчет.

Теперь заголовки Salesperson и Customer появляются наверху каждой страницы. Это по-прежнему не позволяет нам узнать, к какому продавцу относятся клиенты. Нам здесь необходим раздел, который будет печататься наверху только тех страниц, которые являются продолжениями предыдущей страницы и показывают, какой продавец в настоящее время отображен.

Именно это позволяет сделать TransHeader. Прежде всего, необходимо определить то, какому элементу данных соответствует раздел TransHeader. Вы должны спросить себя: «Какой элемент данных отчет выводит на печать, когда он переходит к следующей странице (где я хочу увидеть TransHeader)?». Ответ, в данном случае, – элемент данных клиента.

1 Щелкните раздел области данных Customer.

2 Вставьте раздел TransHeader для элемента данных клиента. Обратите внимание, что он размещается между разделами заголовка и области данных. В разделе Customer TransHeader вы должны отобразить текущего продавца.

3 Нажмите раздел Salesperson Body и выберите поля **Code** и **Name**.

4 Нажмите Edit, Copy, чтобы скопировать эти элементы управления в буфер обмена.

5 Выберите новый раздел TransHeader.

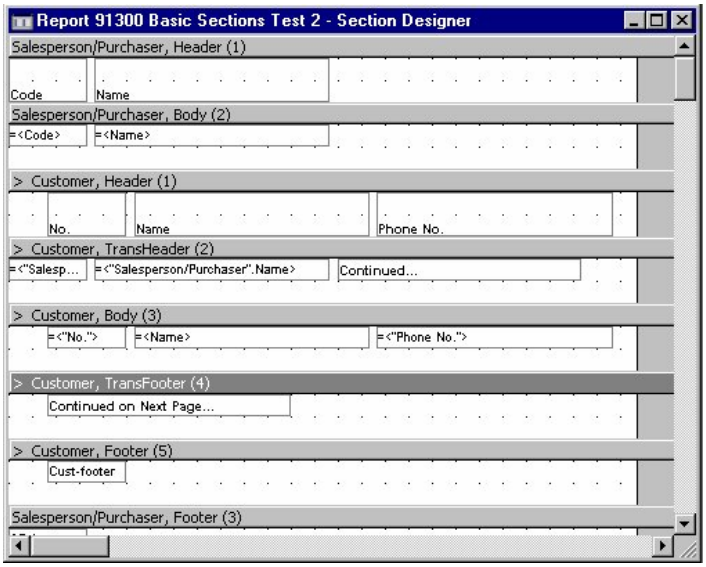
6 Нажмите Edit, Paste, чтобы вставить эти два элементы управления в данный раздел. Выровняйте их с теми, что вы

скопировали. Это почти сработает. Поскольку мы теперь находимся в разделе элемента данных клиента, мы не можем отображать поля из другого элемента данных, не определив этот элемент данных.

7 Выберите текстовую зону Code и откройте окно **Properties**.

8 Измените значение свойства SourceExpr на *Salesperson/Purchaser.*Code. Вы теперь определили другой элемент данных, частью которого фактически является Code.

9 Повторите этот процесс по отношению к текстовой зоне Name. Значением свойства SourceExpr должно быть *"Salesperson/Purchaser".Name.* Кроме того, вам потребуется добавить надпись, содержащую *Continued* в заголовке, в этот раздел.



10 Закройте, сохраните и откомпилируйте отчет.

11 Откройте отчет.

Где напечатан раздел TransHeader (он будет не на каждой странице)?

Transfooter

Раздел TransFooter функционирует в основном так же, как TransHeader, но он печатается в нижней части той страницы, печать которой должна продолжиться на следующей.

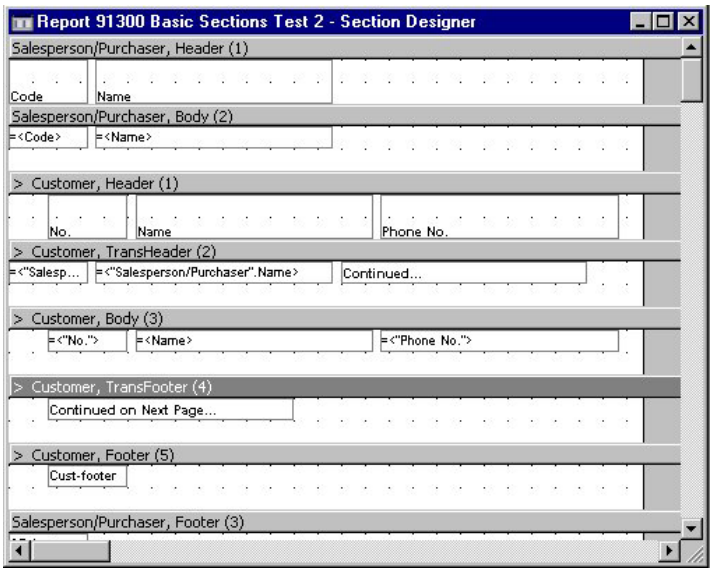
1 Откройте отчет 91300 в Дизайнере отчетов.

2 Нажмите кнопку View, Sections.

3 Выберите раздел области данных Customer.

4 Вставьте раздел TransFooter для элемента данных Customer.

5 Добавьте надпись *Continued on Next Page…* в свойство Caption.



6 Закройте, сохраните и откомпилируйте отчет.

7 Откройте отчет.

Где напечатан раздел TransFooter (он будет не на каждой странице)?

4.4 НЕКОТОРЫЕ ПОЛЕЗНЫЕ СВОЙСТВА

Упражнения, приведенные в этом разделе, помогут вам понять некоторые полезные свойства. Вы будете постепенно улучшать отчет, который вы создали в предыдущих упражнениях.

Упражнение 4.4.1 Свойство NewPagePerRecord

Пользователь хотел бы, чтобы отчет Customer by Salesperson продолжался на новой странице каждый раз, когда обнаруживается новый продавец. Именно для этого предназначено свойство NewPagePerRecord.

1 Откройте отчет 91300 в Дизайнере отчетов.

2 Выберите элемент данных Salesperson/Purchaser.

3 Измените значение свойства NewPagePerRecord на *Yes*.

4 Закройте, сохраните и откомпилируйте отчет.

5 Откройте отчет.

В каких случаях теперь создается новая страница (это происходит дважды)?

Упражнение 4.4.2 Свойство PrintOnlyIfDetail

Обратите внимание, что отчет не содержит каких-либо клиентов для продавца Alice Hart. На самом деле, она – не продавец. Она – покупатель. Мы должны изменить отчет так, чтобы он содержал только тех продавцов, которые имеют клиентов. Помните, что в этом отчете клиент – элемент данных “детали” (detail).

1 Откройте отчет 91300 в Дизайнере отчетов.

2 Выберите элемент данных Salesperson/Purchaser. Это элемент данных “основа” (master). Элемент данных клиент – это элемент данных “детали”. В отчете должны печататься только главные записи, которые содержат детали.

3 Измените свойство PrintOnlyIfDetail на *Yes*.

4 Закройте, сохраните и откомпилируйте отчет.

5 Откройте отчет.

Какие продавцы содержатся в отчете?

Упражнение 4.4.3 ReqFilterFields и ReqFilterHeading

Вы как пользователь хотели бы, чтобы отчет позволил фильтровать поля Code и Name таблицы Salesperson/Purchaser в форме запроса. Свойство ReqFilterFields позволяет вам определять, какие поля пользователь может видеть на форме запроса.

1 Откройте отчет 91300 в Дизайнере отчетов.

2 Выберите элемент данных Salesperson/Purchaser.

3 Измените значение свойства ReqFilterFields на *Code,Name*.

4 Закройте, сохраните и откомпилируйте отчет.

5 Откройте отчет.

Какие поля появились на вкладке **Salesperson/Purchaser** формы запроса?

Можете ли вы фильтровать какие-либо другие поля?

Что еще эта вкладка позволяет вам делать с элементом данных?

Возможно, вам потребуется изменить имя вкладки (или Heading) из *Salesperson/Purchaser* на *Salesperson* в форме запроса. Свойство ReqFilterHeading позволяет вам изменить имя вкладки в форме запроса.

1 Откройте отчет 91300 в Дизайнере отчетов.

2 Выберите элемент данных Salesperson/Purchaser.

3 Измените свойство ReqFilterHeading на *Salesperson*.

4 Закройте, сохраните и откомпилируйте отчет.

5 Откройте отчет.

Какое имя появляется на вкладке?

Примечание

Сортировка элемента данных, который не имеет ReqFilterFields, приводит к исчезновению закладки в форме запроса.

4.5 Группировка в отчете

В этом разделе вы изучите, как группировать данные в отчете.

Что такое группировка?

В отчете вам обычно придется печатать промежуточные суммы или отделять группы записей на основе полей в самих записях. В системе Navision вы должны сообщить отчету, какие поля вы бы хотели контролировать на предмет возникновения этих событий. Отчет контролирует поля, и когда их значение изменяется, это вызывает вывод на печать специальных разделов в отчете. Эти специальные разделы не начинают печататься, если отчет не группирует по полям.

Имейте в виду, что группировка (grouping) не обязательна, если вы можете использовать отступ, чтобы связывать и фильтровать свои элементы данных так, как хотите. Группировка должна использоваться в тех случаях, когда поле, по которому вы хотите группировать, не связано с другой таблицей.

Наиболее обычное использование группировки – находить промежуточную сумму по дате (Date). Поля дат не связаны с таблицей, но обозначают важные группы в таблице. Например, вы хотите увидеть бухгалтерские записи Customer Ledger Entries по клиенту, а для каждого Клиента – по времени. Все, что вам нужно сделать – это группировать по одному полю – дате. Вы можете "группировать" записи по клиенту, используя отступ. (Термин заключен в кавычки, потому что, хотя он и правилен, здесь мы фактически говорим не о группировке отчета.)

Примечание

В большинстве случаев легче использовать отступ, чем группировку. Группировка более чем в одном поле не так проста и предполагает написание кода. Отступ – более естественный способ использования Дизайнера отчетов (если вы можете).

Свойство GroupTotalFields

Это свойство является ключевым при печати специальных разделов. Вы должны определить поля, которые отчет по вашему желанию должен отслеживать (или разрывать) в этом свойстве, чтобы группировать отчет. Это свойство не имеет никакого отношения к Totals или отступу. Имя свойства может ввести в заблуждение.

Чтобы должным образом использовать это поле, вы должны выполнить два действия.

- Прежде всего, поля, которые вы определяете в этом свойстве (их может быть несколько) должны также быть в ключе, который вы определяете в свойстве DataItemTableView. Если вы не выбираете соответствующий ключ, свойство GroupTotalFields, которое вы определяете, будет в отчете игнорироваться.
- Во-вторых, вы должны создать эти специальные разделы. Единственное, что делает группировка, – так это печатает эти разделы в нужное время.

Разделы GroupHeader и GroupFooter

Группировка отчета позволяет вам печатать разделы GroupHeader и GroupFooter. Как можно догадаться по названиям, они печатаются до того, как группа будет запущена и после того, как группа завершена. Раздел GroupHeader печатается после всех разделов Header и перед разделами области данных для этой группы (или сразу после раздела GroupFooter предыдущей группы). Раздел GroupFooter печатается после последнего раздела области данных для этой группы и перед разделом GroupHeader следующей группы (или перед разделом нижнего колонтитула).

Упражнение 4.5.1 Как работает группировка

В этом упражнении вы создадите отчет с одним элементом данных и используете группировку для формирования разрывов в отчете.

1. Откройте Дизайнер объектов.
2. Щелкните Report.
3. Нажмите кнопку New, и появится окно **New Report**. Вы не будете использовать мастер Report, хотя он может использоваться для создания отчета.
4. Нажмите кнопку OK, и появится Дизайнер отчетов.
5. Выберите первую пустую строку в Дизайнере отчетов и введите номер или имя таблицы **Cust. Ledger Entry** (21).
6. Измените свойство GroupTotalFields на *Posting Date.*
7. Нажмите кнопку View, Sections.
8. Вставьте раздел Header.

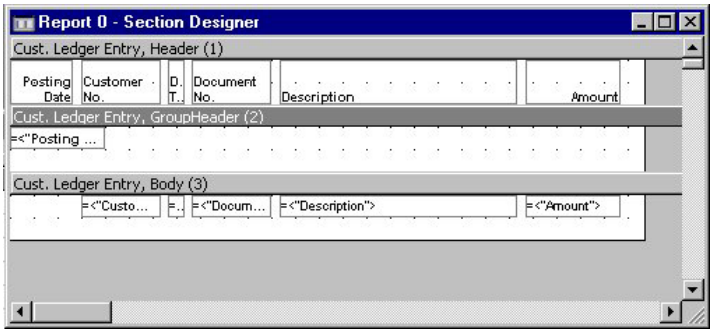
9 Вставьте раздел Group Header.

10 В разделе области данных добавьте поля **Posting Date**, **Customer No.**, **Document Type**, **Document No.**, **Description** и **Amount**.

11 Переместите надписи наверх, в раздел Header.

12 Выберите текстовую зону Posting Date и перетащите ее из раздела области данных в раздел Group Header.

13 Измените значение свойства FontBold на *Yes*.



14 Закройте, сохраните и откомпилируйте отчет с идентификатором *91500* и именем *Customer Detail*.

15 Откройте отчет.

Где был напечатан раздел GroupHeader?

Почему он не был напечатан?

Создание необходимого ключа

Помните, что вы должны сортировать элемент данных, используя поле (поля), которое вы хотите сгруппировать. К сожалению, в таблице **Cust. Ledger Entry** нет ключа, который начинается с поля **Posting Date**. Поэтому вы создадите новый ключ в соответствующей таблице. Возможно, вам придется делать это во многих создаваемых вами отчетах.

1 Откройте таблицу 21 в окне Table Designer.

2 Нажмите кнопку View, Keys.

3 Выберите пустую строку внизу окна **Key Designer**.

4 Введите *Posting Date* в столбец Key.

5 Закройте окно **Key Designer**.

6 Закройте, сохраните и откомпилируйте таблицу.

Завершение

Теперь вы можете закончить свой отчет.

1 Откройте отчет 91500 в Дизайнере отчетов.

2 Выберите элемент данных и откройте окно **Properties**.

3 Измените значение свойства DataItemTableView на *SORTING(Posting Date)*. Вы можете использовать кнопку поясняющего редактирования, чтобы открыть окно **Table View** и выбрать новый ключ, который вы создали.

4 Закройте, сохраните и откомпилируйте отчет.

5 Откройте отчет.

Где был напечатан раздел GroupHeader?

Упражнение 4.5.2 Как могут взаимодействовать группировка и определение отступа

В этом упражнении вы будете изучать, как группировка и определение отступа могут взаимодействовать для группирования данных на множестве различных уровней.

1 Откройте отчет 91500 в Дизайнере отчетов.

2 Нажмите File, Save As.

3 Введите идентификатор *91501* и имя *Customer Detail 2*.

4 Нажмите кнопку OK для сохранения отчета с новым идентификатором и именем.

Теперь вы измените этот отчет так, чтобы он группировал Customer Ledger Entries сначала по Customer, а затем по дате регистрации. Вместо того чтобы пытаться добавить Customer No. в свойство GroupTotalFields, вы будете использовать отступ, чтобы группировать по клиенту.

5 Вставьте (F3) строку над элементом данных Cust. Ledger Entry.

6 Выберите таблицу Customer для этого нового элемента данных.

7 Выделите отступом элемент данных Cust. Ledger Entry один раз под элементом данных Customer.

8 Измените значение свойства DataItemLink для элемента данных Cust. Ledger Entry на *“Customer No.” = FIELD(“No.”).* Вы можете использовать кнопку поясняющего редактирования, чтобы облегчить себе задачу установки этого свойства.

Помните, что вы всегда должны использовать соответствующий ключ для элементов данных, выделенных отступом. Для этого вам нужно, чтобы записи регистрации сортировались по Customer No., а затем по Posting Date. Есть ключ, связанный с этими полями.

9 Измените свойство DataItemTableView , чтобы сортировать по ключу, который содержит Customer No., а затем Posting Date.

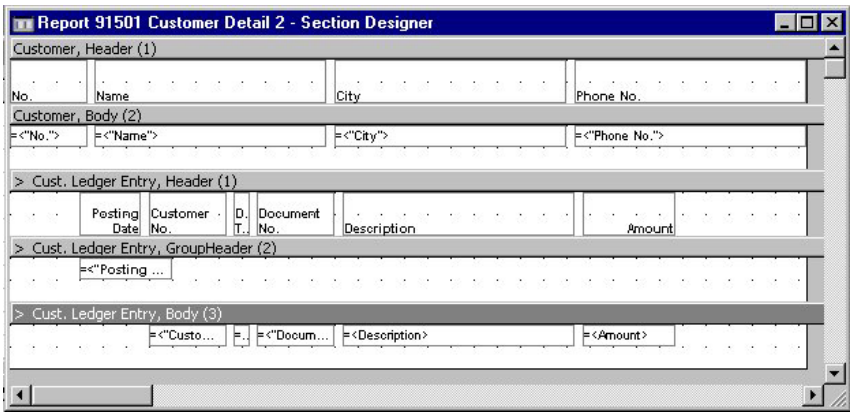
10 Нажмите File, Save, чтобы сохранить отчет.

11 Нажмите кнопку View, Sections.Теперь вы должны вставить раздел Header для элемента данных Customer.

12 В разделе области данных Customer добавьте поля **No.**, **Name**, **City**, и **Phone No.**

13 Переместите надписи наверх, в раздел Header.

Возможно, вам потребуется переместить элементы управления в разделах Cust. Ledger Entry наверх вправо, чтобы показать отступ, который мы используем. Наверное, вы заметили символ «>» в серой рамке разделов Ledger. Он показывает, что этот элемент данных выделен отступом. Чтобы сделать это, просто выделите все элементы управления в разделе и переместите их вправо, как вы сделали бы это на форме.



14 Закройте, сохраните и откомпилируйте отчет.

15 Выполните отчет, и появится форма запроса.

16 Нажмите кнопку Preview.

Где был напечатан раздел GroupHeader?

Где напечатаны разделы Header для каждого элемента данных?

4.6 СУММИРОВАНИЕ В ОТЧЕТЕ

В этом разделе вы изучите, как добавить итоговую сумму к разделам нижнего колонтитула в отчете. Как вы увидите в упражнениях, это очень легко.

Что такое суммирование?

Почти всегда в финансовом пакете отчет предназначен для суммирования некоторых данных, находящихся в БД. Это обычно предполагает суммирование данных одного или нескольких полей, которые отображаются или считываются отчетом, а затем – отображение общей суммы. Мы называем этот процесс «суммированием» отчета (totaling).

В системе Navision используются суммы промежуточные Subtotals и итоговые Grandtotals. Промежуточные суммы Subtotals следуют после логической группировки записей и показывают сумму значений группы. Итоговые суммы Grandtotals печатаются в самом конце отчета и показывают сумму для всего отчета (всех промежуточных сумм).

Свойство TotalFields

Чтобы отчет вывел сумму, вы должны ему сообщить, какие поля он должен сложить (суммировать). Свойство TotalFields элемента данных позволяет вам определять поля, данные которых вы хотите суммировать. Называя поле в этом свойстве, вы инструктируете отчет отслеживать суммы данных этого поля на протяжении всего отчета. Это, однако, не означает, что отчет должен отображать сумму.

Как отобразить сумму?

Чтобы отобразить общие количества для полей, которые вы определили в свойстве TotalFields, необходимо выполнить два действия.

- Сначала создайте раздел для отображения суммы. В отчете будут отображаться только общие количества в каком-то типе раздела Footer (Footer, GroupFooter, TransFooter или TransHeader).
- Во-вторых, поднимите вверх меню полей для этого раздела и опустите поля, которые вы складываете, в этот раздел. Вам может потребоваться удалить надписи и выровнять элементы управления по отношению к подходящим столбцам. Именно

так. Поскольку вы отображаете эти поля в типе раздела Footer, вместо показа текущего значения полей отчет отображает общие суммы.

Упражнение 4.6.1 Создание Subtotals и Grand Totals

В этом упражнении вы добавите промежуточные и итоговые суммы в один из отчетов, который вы создали в разделе 4.5.

1 Откройте отчет 91500 в Дизайнере отчетов.

2 Выберите элемент данных Cust. Ledger Entry.

3 Откройте окно **Properties**.

4 Измените свойство TotalFields на *Amount*. Это позволяет отчету вычислить общую сумму для поля **Amount**, при обработке этого элемента данных.

5 Нажмите кнопку View, Sections.

6 Вставьте раздел GroupFooter и раздел Footer.

7 Откройте окно **Field Menu**.

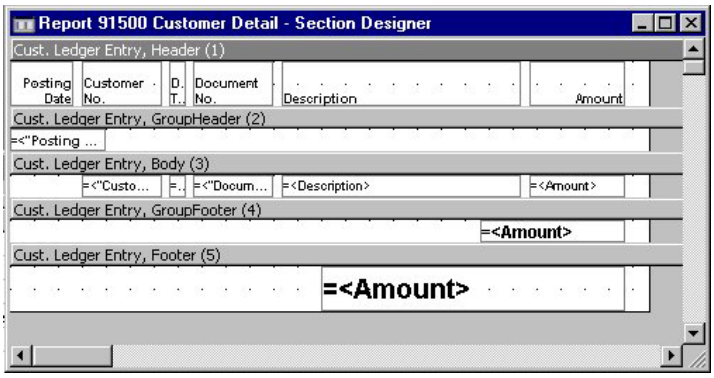
8 Выберите поле **Amount**.

9 Добавьте это поле в разделу Group Footer.

10 Удалите надпись.

11 Выровняйте поле **Amount** по отношению к столбцу количества. Вам может потребоваться изменить шрифт или сделать его полужирным.

12 Повторите этот процесс в отношении раздела Footer.



13 Закройте, сохраните и откомпилируйте отчет.

14 Откройте отчет. Появится форма запроса .

15 Нажмите кнопку Preview.

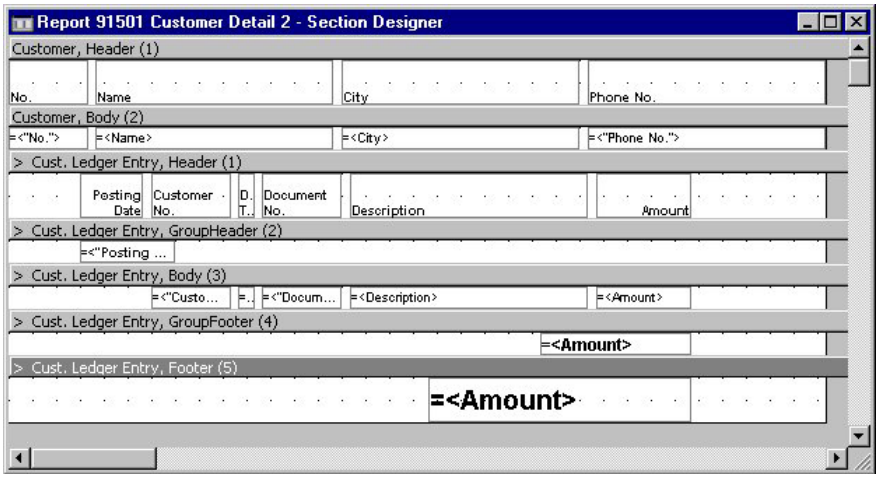
Верна ли сумма, как вам кажется?

Как вы сообщили системе, что поле Amount в разделе GroupFooter должно показать сумму только для этой группы?

Упражнение 4.6.2 Добавление Subtotals и Grand Totals к выделенным отступом элементам данных

В этом упражнении вы добавите промежуточные суммы к своему отчету, содержащему отступ и группировку.

1. Откройте отчет 91501 в Дизайнере отчетов.
2. Выберите элемент данных Cust. Ledger Entry.
3. Откройте окно **Properties**.
4. Измените свойство TotalFields на *Amount*. Это сообщает отчету вычислять общую сумму для поля **Amount**, при обработке этого элемента данных (в отчете вычисляются только суммы для одного элемента данных в каждый момент времени).
5. Нажмите кнопку View, Sections.
6. Вставьте разделы GroupFooter и Footer для элемента данных Cust. Ledger Entry.
7. Откройте окно **Field Menu**.
8. Выберите поле **Amount** и добавьте это поле в раздел GroupFooter.
9. Удалите надпись.
10. Выровняйте поле **Amount** по отношению к столбцу количества. Вам может потребоваться изменить шрифт или сделать его полужирным.
11. Повторите этот процесс в отношении раздела Footer.



12 Закройте, сохраните и откомпилируйте отчет.

13 Откройте отчет. Появится форма запроса .

14 Нажмите кнопку Preview.

Верна ли сумма, как вам кажется?

Какая сумма печатается в разделе GroupFooter?

Какая сумма печатается в разделе Footer?

Можно ли добавить GrandTotal в этот отчет, не добавляя код?

4.7 Добавление расширенных возможностей

В этом разделе вы познакомитесь с некоторыми интересными возможностями отчетов. Для некоторых из этих особенностей требуется написать пару строк кода, но вы получите точные инструкции, где именно разместить код.

Вы узнаете, как использовать FlowFilters в отчетах, изменять форму запроса, пропускать разделы и находить итоговые суммы нескольких элементов данных.

Упражнение 4.7.1 Использование FlowFilters в отчете

В этом упражнении вы добавите возможность фильтровать отчет с полем **Date Filter** в таблице **Custome**r. Это упражнение зависит от упражнений, представленных в разделе 4.6. Вы должны выполнить те упражнения, прежде чем приступать к этим.

1 Откройте отчет 91501 в Дизайнере отчетов.

2 Выберите элемент данных Customer.

3 Измените ReqFilterFields, включив поле **Date Filter**.

4 Закройте, сохраните и откомпилируйте отчет.

5 Откройте отчет. Появится форма запроса .

6 В столбце Filter поля Date Filter введите диапазон чисел *01/01/01..12/31/01*.

7 Нажмите кнопку Preview.

Есть ли в отчете записи, выходящие за пределы этого диапазона?

Почему вы думаете, что отчет не изменился?

8 Сконструируйте отчет 91501.

9 Выберите элемент данных Cust. Ledger Entry.

10 Измените значение свойства DataItemLink на *Customer No.=FIELD(No.),Posting Date=FIELD(Date Filter)*. Вы можете использовать кнопку поясняющего редактирования, чтобы облегчить себе задачу изменения этого свойства.

11 Закройте, сохраните и откомпилируйте отчет.

12 Откройте отчет. Появится форма запроса .

13 В столбце Filter поля Date Filter введите диапазон чисел *01/01/01..12/31/01*.

14 Нажмите кнопку Preview.

Есть ли в отчете записи, выходящие за пределы этого диапазона?

Что происходит с отчетом, если вы оставляете Date Filter на форме запроса пустым?

Упражнение 4.7.2 Добавление и удаление из формы запроса

В этом упражнении вы добавите флажок на вкладку **Options** формы запроса и удалите другие вкладки с формы запроса.

Появляющиеся и исчезающие вкладки

Возможно, вы заметили, что не каждый элемент данных имеет вкладку (на форме запроса) в созданных вами отчетах. Фактически каждый элемент данных содержит вкладку, но, в зависимости от определенных свойств, вкладка может появляться или не появляться на форме запроса.

1 Выполните отчет 91501. Появится форма запроса.

Какие вкладки появляются на форме запроса?

2 Откройте отчет 91501 в Дизайнере отчетов.

3 Выберите элемент данных Cust. Ledger Entry.

4 Удалите свойство DataItemTableView, чтобы снова установить значение по умолчанию.

5 Закройте, сохраните и откомпилируйте отчет.

6 Откройте отчет. Появится форма запроса.

Какие вкладки имеются сейчас на форме запроса?

Что пользователь может делать на каждой вкладке (обратите внимание на кнопку)?

7 Откройте отчет 91501 в Дизайнере отчетов.

8 Выберите элемент данных Cust. Ledger Entry.

9 Измените значение свойства DataItemTableView на *SORTING(Customer No.,Posting Date,Currency Code)*.

10 Закройте, сохраните и откомпилируйте отчет.

11 Откройте отчет. Появится форма запроса.

Какие вкладки имеются сейчас на форме запроса?

12 Откройте отчет 91501 в Дизайнере отчетов.

13 Выберите элемент данных Cust. Ledger Entry.

14 Измените значение свойства ReqFilterFields на *Posting Date*

15 Закройте, сохраните и откомпилируйте отчет.

16 Откройте отчет. Появится форма запроса.

Какие вкладки имеются сейчас на форме запроса?

Что может пользователь делать на вкладке **Cust. Ledger Entry**?

Какие два свойства определяют, видит ли пользователь вкладку для элемента данных на форме запроса?

Добавление флажка на вкладку Options

Чтобы добавить элемент управления данными на вкладку **Options** формы запроса, вы сначала должны создать глобальную переменную, с которой будет связан элемент управления. Пользователь изменит значение переменной, используя элемент управления. Вы будете использовать значение переменной в следующем упражнении.

1 Откройте отчет 91501 в Дизайнере отчетов.

2 Щелкните View, C/AL Globals. Появится окно **C/AL Globals**.

3 Добавьте переменную с именем ShowDetails и установите тип данных *Boolean*.

4 Закройте окно **C/AL Globals**.

5 Щелкните View, Request Form, и появится Request Form Options Designer.

6 Добавьте флажок и надпись (используя кнопку Add Label) на форму.

7 Измените свойство заголовка флажка на *Show Details*. Это изменит заголовок присоединенной надписи.

8 Измените свойство SourceExpr флажка на ShowDetails.

9 Закройте Request Form Options Designer.

10 Закройте, сохраните и откомпилируйте отчет.

11 Откройте отчет. Появится форма запроса.

Какие вкладки имеются сейчас на форме запроса?

Можете ли вы изменить значение флажка, который вы добавили?

Какое значение имеет переменная, когда флажок выставлен?

Какое значение имеет переменная, когда флажок снят?

Выставьте флажок и сделайте предварительный просмотр отчета. Закройте окно предварительного просмотра и повторно запустите отчет. Флажок все еще выставлен?

Каждый раз при запуске отчета переменная устанавливается в значение по умолчанию False. Чтобы сохранить свое значение переменной, вы должны сообщить форме запроса сохранять значения переменных в ZUP-файле.

12 Откройте отчет 91501 в Дизайнере отчетов.

13 Щелкните View, Request Form. Появится Request Form Options Designer.

14 Откройте окно **Properties** этой формы.

15 Измените свойство SaveValues на *Yes*.

16 Закройте Request Form Options Designer.

17 Закройте, сохраните и откомпилируйте отчет.

18 Откройте отчет. Появится форма запроса.

Выставьте флажок и сделайте предварительный просмотр отчета. Закройте окно предварительного просмотра и повторно запустите отчет. Флажок все еще выставлен?

Упражнение 4.7.3 Пропуск разделов в отчете

В этом упражнении вы будете использовать значение переменной, которую добавили в предыдущем упражнении, чтобы отображать или не отображать раздел в напечатанном отчете. Раздел, который вы покажете, носит название Cust. Ledger Entry Body.

1 Откройте отчет 91501 в Дизайнере отчетов.

2 Нажмите кнопку View, Sections. Появится Sections Designer.

3 Щелкните серый прямоугольник раздела Cust. Ledger Entry Body. Это позволяет выбрать данный раздел.

4 Щелкните View, C/AL Code (F9). Появится Редактор C/AL-кода.

5 Введите следующий код в OnPreSection Trigger:
CurrReport.SHOWOUTPUT(ShowDetails);

6 Закройте Редактор C/AL-кода и Дизайнер разделов.

7 Закройте, сохраните и откомпилируйте отчет.

8 Откройте отчет. Появится форма запроса .

9 Убедитесь, что флажок на вкладке **Options** выбран .

10 Нажмите кнопку Preview.

Отчет выглядит так же, как в последний раз, когда вы запускали отчет?

Закройте окно предварительного просмотра. Повторно запустите отчет и убедитесь, что флажок на вкладке **Options** не выставлен.

Что изменилось в отчете?

Упражнение 4.7.4 Добавление итоговой суммы с несколькими элементами данных

В этом упражнении вы добавите итоговую сумму в отчет Customer Detail 2. Это невозможно без написания определенного кода. В данном случае вы добавите только одну строку кода к отчету и, конечно, к новому разделу.

1 Откройте отчет 91501 в Дизайнере отчетов.

2 Выберите элемент данных Customer.

3 Щелкните View, C/AL Code (F9). Появится Редактор C/AL-кода.

4 Введите следующий код в OnPreDataitem Trigger:
CurrReport.CREATETOTALS("Cust. Ledger Entry".Amount);

5 Закройте Редактор C/AL-кода.

6 Нажмите кнопку View, Sections.

7 Вставьте раздел Footer для элемента данных Customer.

8 В этом разделе нижнего колонтитула вручную добавьте текстовую зону.

9 Измените значение свойства SourceExpr текстовой зоны на “*Cust. Ledger Entry".Amount*. Кроме того, вам может потребоваться увеличить размер шрифта или выделить флажок полужирным.

10 Закройте Дизайнер разделов.

11 Закройте, сохраните и откомпилируйте отчет.

12 Откройте отчет. Появится форма запроса .

13 Убедитесь, что флажок на вкладке **Options** выбран .

14 Нажмите кнопку Preview.

Кажется ли вам корректной общая сумма, указанная в конце отчета?

Как еще вы могли бы получить ее?

Упражнение 4.7.5 Добавление в отчет многоязычных функциональных возможностей

Каждый заголовок на форме запроса или в разделах должен содержать заполненное свойство CaptionML и OptionCaptionML. Если это не так, пользователь будет видеть одни заголовки по-английски, а другие – на языке, выбранном в отчете и форме запроса. В связанных элементах управления это свойство уже заполнено. Однако при появлении любых новых полей, которые должны содержать переменные, или вычисленных полей, свойство OptionML также должно быть заполнено. См. главу “Основные таблицы” для получения дополнительной информации о том, как заполнить это свойство.

4.8 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 элемент данных, являющийся внесенной в список таблицей.

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» или, иногда, «Listing».

Примеры

- Перечень клиентов;
- Список товаров.

Отчеты Test

Отчет Test – это отчет, выведенный на печать из таблицы Journal.

Его назначение состоит в том, чтобы проверить каждую Journal Lines согласно тем критериям, которые будут использоваться для Posting, с тем чтобы можно было обнаружить и исправить все ошибки до регистрации.

При обнаружении ошибки во время регистрации обработка останавливается, и ошибка должна быть исправлена перед продолжением Posting.

Таким образом, использование книги учета со множеством ошибок и отчета Test – хороший способ обнаружения этих ошибок.

Именем обычно становится имя соответствующей формы Journal, за которым следует слово «Test».

Примеры

- Общая книга учета – Test;
- Книга учета ресурсов – Test;
- Книга учета платежей Test Report.

Отчеты Posting

Отчет Posting можно печатать как часть опции «зарегистрировать и распечатать» в Journal.

Фактически это отчет, который выводится на печать из Register, и он имеет то же имя, что и Register.

В нем перечислены все сделки (то есть Ledger Entries), которые были зарегистрированы в этом Register.

Примеры

- G/L Register
- Item Register
- Payroll Register

Отчеты Transaction

Отчет Transaction содержит два элемента данных.

Первый – это таблица Master, второй - соответствующая таблица Ledger.

Обычно отчет Transaction содержит список всех регистрационных записей в таблице Ledger.

Как правило, дается промежуточная сумма для каждого отчета 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 Detail/Summary
- Vendor Account Detail
- Employee Transaction Detail

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

- Item Status by Salesperson
- Vendor 1099 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, где информация заголовка напечатана так, как при заполнении счет-фактуры, где этот заголовок повторяется наверху каждой страницы и где ни одна страница не содержит информацию больше чем из одного заголовка.

Строки для счета-фактуры выводятся на печать скорее как обычный отчет, состоящий из строк и столбцов.

Строки соответствуют заголовку на той же странице, а строки из других счетов-фактур не отображаются на той же странице.

Примеры

- Sales Invoice
- Customer Statement
- Purchase Order

4.9 СОЗДАНИЕ ОСНОВНОГО ОТЧЕТА

В этом упражнении вы проверите свою способность создавать простой отчет. В этом разделе нет пошаговых инструкций. Попробовав выполнить его самостоятельно, вы можете затем прочитать описание пошаговых инструкций в разделе «Вопросы и ответы».

Ниже приведены требования к вашему отчету. При создании отчета используйте идентификатор *91700* и имя *Vendor Detail* .

Создайте отчет, отвечающий следующим требованиям.

- На печать выводятся Vendor Ledger Entries по Vendor (два элемента данных, используйте выделение отступом).
- Отображается No., Name и Phone No. из таблицы **Vendor**.
- Отображается Posting Date, Document Type, Document No., Description и Amount из таблицы **Vendor Ledger Entry**.
- Отображается промежуточная сумма в поле **Amount** для Vendor.

Измените этот отчет так, чтобы он включал другую промежуточную сумму, например, промежуточную сумму для каждой уникальной даты регистрации, относящейся к определенному поставщику. Для этого потребуется группировка в поле Posting Date. Не забудьте выбрать подходящий ключ.

Измените отчет так, чтобы в нем не отображались поставщики, у которых нет каких-то записей деталей.

Измените отчет так, чтобы в нем не отображались какие-либо отчеты Ledger, тип Document которых – Credit Memo. Для этого потребуется фильтрация этого элемента данных. Помните, что слово View означает Sorting и Filtering.

4.10 Ответы на вопросы

Упражнение 4.2.1 Создание модели данных

Здесь нет вопросов.

Упражнение 4.2.2 Формирование композиции страницы

Отчет выглядит так, как вы ожидали? Где печатаются разделы заголовка для каждого элемента данных?

Да.

Раздел Header для элемента данных Salesperson печатается только наверху первой страницы.

Раздел Header для элемента данных Customer печатается после каждого раздела области данных Salesperson, если у этого Salesperson есть клиенты. Он не распечатывается для Salesperson, если клиентов нет.

Повторялись ли разделы заголовка?

Да, раздел Header для элемента данных Customer повторяется для каждого Salesperson, который имеет клиентов.

Упражнение 4.3.1 Заголовки, области данных и нижние колонтитулы

Как отчет прошел цикл через таблицы?

Сначала он прошел цикл через все Salespeople. Затем он прошел цикл через все Customers (останавливаясь на записи пятого клиента).

Где печатается раздел Header каждого элемента данных?

Прямо перед тем, как он начинают выводиться на печать разделы области данных. Они выводятся на печать только один раз.

Где печатается раздел Footer каждого элемента данных?

Сразу после всех разделов области данных. Они выводятся на печать только один раз.

С отступом

Как отчет прошел цикл через таблицы?

Сначала он прошел цикл через все Salespeople, но для каждой записи salesperson он прошел цикл через таблицу Customer (останавливаясь на записи пятого клиента).

Где печатается раздел Header каждого элемента данных?

Прямо перед тем, как он начинают выводиться на печать разделы области данных. Раздел заголовка продавца выводится на печать только один раз наверху первой страницы. Раздел Header клиента выводится на печать сразу после каждого раздела области данных продавца.

Где печатается раздел Footer каждого элемента данных?

Сразу после всех разделов области данных. Раздел нижнего колонтитула клиента выводится на печать несколько раз в отчете прямо перед разделом области данных (или нижнего колонтитула) следующего продавца. Раздел нижнего колонтитула продавца выводится на печать один раз в самом конце отчета.

Со связыванием

Как отчет прошел цикл через таблицы?

Сначала он прошел цикл через все Salespeople, но для каждой записи продавца он прошел цикл через Customers для этих продавцов (если они были, при этом останавливаясь на записи пятого клиента).

Где печатается раздел Header каждого элемента данных?

Прямо перед тем, как он начинают выводиться на печать разделы области данных. Раздел заголовка продавца выводится на печать только один раз наверху первой страницы. Раздел Header клиента выводится на печать сразу после каждого раздела области данных продавца (если у продавца есть клиенты).

Где печатается раздел Footer каждого элемента данных?

Сразу после всех разделов области данных. Раздел нижнего колонтитула клиента выводится на печать несколько раз в отчете прямо перед разделом области данных (или нижнего колонтитула) следующего продавца. Раздел нижнего колонтитула продавца выводится на печать один раз в самом конце отчета.

Упражнение 4.3.2 TransHeaders и TransFooters

Где напечатан раздел TransHeader (он будет не на каждой странице)?

Наверху второй страницы после Salesperson Header и Customer Header.

Transfooter

Где напечатан раздел TransFooter (он будет не на каждой странице)?

В самом низу первой страницы.

Упражнение 4.4.1 Свойство NewPagePerRecord

В каких случаях теперь создается новая страница (это происходит дважды)?

Когда отчет продолжается на следующей странице и когда элемент данных Salesperson переходит в следующую запись.

Упражнение 4.4.2 Свойство PrintOnlyIfDetail

Какие продавцы содержатся в отчете?

JR и PS

Упражнение 4.4.3 ReqFilterFields и ReqFilterHeading

Какие поля появились на вкладке **Salesperson/Purchaser** формы запроса?

Code и Name

Можете ли вы фильтровать какие-либо другие поля?

Да. Я могу щелкнуть в пустой строке и добавить любое другое поле из таблицы.

Что еще эта вкладка позволяет вам делать с элементом данных?

Сортировать его. Я могу щелкнуть кнопку сортировки и изменить ключ, используемый этим отчетом.

Какое имя появляется на вкладке?

Salesperson

Упражнение 4.5.1 Как работает группировка

Где был напечатан раздел Group Header?

Он не был напечатан в отчете.

Почему он не был напечатан?

Не была произведена сортировка элемента данных с помощью ключа, содержащего Posting Date.

Создание необходимого ключа

Завершение

Где был напечатан раздел Group Header?

Перед разделами области данных, содержавшими новую дату.

Упражнение 4.5.2 Как могут взаимодействовать группировка и определение отступа

Где был напечатан раздел GroupHeader?

Перед разделами области данных, содержавшими новую дату.

Где напечатаны разделы Header для каждого элемента данных?

Раздел заголовка клиента был выведен на печать наверху первой страницы. Раздел заголовка гроссбуха клиента был выведен на печать сразу после каждого раздела области данных клиента (для клиентов, содержавших подробные записи).

Упражнение 4.6.1 Создание Subtotals и Grand Totals

Верна ли сумма, как вам кажется?

Да. Промежуточные суммы больше, чем значения строк, а итоговая сумма больше, чем какая-либо промежуточная сумма.

Как вы сообщили системе, что поле **Amount** в разделе Group Footer должно показать сумму только для этой группы?

Просто установив значение свойства TotalFields. В отчете заменяется значение поля количества значением из суммы количественных полей для данной группы.

Упражнение 4.6.2 Добавление Subtotals и Grand Totals к выделенным отступом элементам данных

Верна ли сумма, как вам кажется?

Да. Промежуточные суммы больше, чем значения строк.

Какая сумма печатается в разделе Group Footer?

Сумма для этой даты.

Какая сумма печатается в разделе Footer?

Сумма для этого клиента.

Можно ли добавить Grand Total в этот отчет, не добавляя код?

Нет. Отчет может отображать только сумму на один элемент данных в каждый момент времени.

Создание основного отчета

Ниже приведены пошаговые инструкции для создания этого отчета.

Этап 1

1 Откройте Дизайнер объектов.

2 Щелкните Report.

3 Нажмите кнопку New. Появится окно **New Report**.

4 Нажмите кнопку OK. Появится Дизайнер отчетов.

5 Выберите первую пустую строку в Дизайнере отчетов и введите номер или имя таблицы **Vendor** (23).

6 Выберите следующую пустую строку в Дизайнере отчетов и введите номер или имя таблицы **Vendor Ledger Entry** (25).

7 Выделите отступом элемент данных *Vendor Ledger Entry,* щелкнув правую кнопку со стрелкой в нижней части формы.

8 При выбранном по-прежнему элементе данных Vendor Ledger Entry извлеките свойства этого элемента данных.

9 Щелкните свойство the *DataItemLinkReference*. Убедитесь в том, что оно установлено на Vendor.

10 Измените значение свойства *DataItemLink* на *DataItemLink*.

11 В главном меню нажмите View, Sections.

12 Вставьте раздел Vendor Header.

13 Измените значение свойства PrintOnEveryPage на *Yes*.

14 Вставьте раздел Vendor Ledger Entry Header.

15 Измените значение свойства PrintOnEveryPage на *Yes*.

16 Нажмите раздел элемента данных в Vendor.

17 Из Field Menu добавьте поля **No.**, **Name** и **Phone No.**

18 Переместите надписи в заголовок.

19 Нажмите раздел области данных в Vendor Ledger Entry.

20 Из Field Menu добавьте поля **Posting Date**, **Document Type**, **Document No.**, **Description** и **Amount**.

21 Переместите надписи в заголовок.

22 Вставьте раздел нижнего колонтитула Vendor Ledger Entry.

23 Добавьте поле Amount в раздел нижнего колонтитула Vendor Ledger Entry. Удалите надпись.

24 Выровняйте поле **Amount** в разделе нижнего колонтитула по отношению к полю **Amount** в области данных.

25 Измените свойство FontBold поля **Amount** в разделе нижнего колонтитула на *Yes*.

26 Сохраните отчет с идентификатором *91700* и именем *Vendor Detai.*

Этап 2

1 Откройте отчет 91700 в Дизайнере отчетов.

2 Выберите элемент данных Vendor Ledger Entry.

3 Измените свойство GroupTotalFields на Posting Date.

4 Измените свойство DataItemTableView на *SORTING(Vendor No.,Posting Date,Currency Code)*.

5 Нажмите кнопку View, Sections.

6 Вставьте раздел нижнего колонтитула группы Vendor Ledger Entry.

7 Добавьте поле **Amount** в раздел нижнего колонтитула группы Vendor Ledger Entry и удалите надпись.

8 Выровняйте поле **Amount** в разделе нижнего колонтитула по отношению к полю **Amount** в области данных.

9 Закройте, сохраните и откомпилируйте отчет.

Этап 3

1 Откройте отчет 91700 в Дизайнере отчетов.

2 Выберите элемент данных Vendor.

3 Измените свойство PrintOnlyIfDetail на *Yes*.

4 Закройте, сохраните и откомпилируйте отчет.

Этап 4

1 Откройте отчет 91700 в Дизайнере отчетов.

2 Выберите элемент данных Vendor Ledger Entry.

3 Измените значения свойства DataItemTableView на *SORTING(Vendor No.,Posting Date,Currency Code) WHERE(Document Type=FILTER(<>Credit Memo))*.

4 Закройте, сохраните и откомпилируйте отчет.

Теперь отчет закончен.

Упражнение 4.7.1 Использование FlowFilters в отчете

Есть ли в отчете записи, выходящие за пределы этого диапазона?

Да. Отчет вообще не был отфильтрован.

Почему вы думаете, что отчет не изменился?

Второй элемент данных (Cust. Ledger Entry) не знает о фильтре, который установил пользователь.

Есть ли в отчете записи, выходящие за пределы этого диапазона?

Нет. Отчет отфильтрован.

Что происходит с отчетом, если вы оставляете Date Filter на форме запроса пустым?

Будут показаны все записи. Фильтры не применяются.

Упражнение 4.7.2 Добавление и удаление из формы запроса

Какие вкладки появляются на форме запроса?

Customer

Какие вкладки имеются сейчас на форме запроса?

Customer, Cust. Ledger Entry.

Что пользователь может делать на каждой вкладке (обратите внимание на кнопку)?

Он может фильтровать или сортировать элемент данных на обеих вкладках.

Какие вкладки имеются сейчас на форме запроса?

Customer

Какие вкладки имеются сейчас на форме запроса?

Customer, Cust. Ledger Entry.

Что может пользователь делать на вкладке **Cust. Ledger Entry**?

Он может фильтровать элемент данных. Кнопка сортировки больше не появляется.

Какие два свойства определяют, видит ли пользователь вкладку для элемента данных на форме запроса?

ReqFilterFields, DataItemTableView. Вкладка покажет, нет ли сортировки и определены ли поля фильтра запроса. Эта вкладка не будет показывать, сортируете ли вы элемент данных, и определяете ли поля фильтра запроса.

Добавление флажка на вкладку Options

Какие вкладки имеются сейчас на форме запроса?

Customer, Cust. Ledger Entry, Options

Можете ли вы изменить значение флажка, который вы добавили?

Да. Его можно установить или снять.

Какое значение имеет переменная, когда флажок выставлен?

True.

Какое значение имеет переменная, когда флажок снят?

Неверно.

Выставьте флажок и сделайте предварительный просмотр отчета. Закройте окно предварительного просмотра и повторно запустите отчет. Флажок все еще выставлен?

Нет.

Выставьте флажок и сделайте предварительный просмотр отчета. Закройте окно предварительного просмотра и повторно запустите отчет. Флажок все еще выставлен?

Да.

Упражнение 4.7.3 Пропуск разделов в отчете

Отчет выглядит так же, как в последний раз, когда вы запускали отчет?

Да. В нем показаны все разделы.

Закройте окно предварительного просмотра. Повторно запустите отчет и убедитесь, что флажок на вкладке **Options** не выставлен.

Что изменилось в отчете?

Подробная информация в разделах Body элемента данных Cust. Ledger Entry больше не отображаются в отчете. Фактически не

отображается весь раздел. Только разделы элемента данных Cust.. Ledger Entry печатаются вместе с разделами Customer.

Упражнение 4.7.4 Добавление итоговой суммы с несколькими элементами данных

Кажется ли вам корректной общая сумма, указанная в конце отчета?

Да.

Как еще вы могли бы получить ее?

Можно было бы прибавить глобальную переменную и самостоятельно сложить записи.

Упражнение 4.7.5 Добавление в отчет многоязычных функциональных возможностей

При создании новых или модификации существующих отчетов необходимо позаботиться о том, чтобы все заголовки данных на форме запроса и разделы содержали CaptionML and OptionCaptionML, как было показано ранее в главе “Основные таблицы”. Свойство Name должно остаться номером элемента управления, например, Control 9.

4.11 Обзор главы

В этом разделе содержатся вопросы, касающиеся материала данной главы. Ответы на вопросы можно найти в подразделе «Ответы».

Вопросы

Верно или неверно следующее утверждение? Все отчеты должны содержать минимум одну страницу.

Перечислите имена трех Дизайнеров, используемых совместно с отчетами, и опишите их действие.

Сколько раз раздел Header печатается в отчете для элемента данных, выделенного отступом (предположим, страница PrintOnEvery выключена, а в таблице есть данные)? Что если элемент данных не выделен отступом?

Какое свойство элемента данных может использоваться для удаления вкладки элементов данных с формы запроса (предположим, поля фильтра запроса не установлены)?

Какие два свойства элемента данных должны быть установлены для вывода на печать промежуточной суммы поля в разделе нижнего колонтитула группы?

Когда раздел TransHeader печатается для элемента данных?

Какое свойство можно установить, чтобы начинать новую страницу каждый раз при считывании записи?

Какие два свойства можно использовать, чтобы напечатать нижний колонтитул на всех страницах?

Дан отчет, содержащий два элемента данных, второй из которых выделен отступом. Какой C/AL-код могли бы вы написать, чтобы он сильнее всего напоминал эту ситуацию?

Какое свойство может использоваться для того, чтобы фильтровать элемент данных, выделенный отступом, на основе полей в не выделенном отступом элементе данных, который находится над ним?

Ответы

Верно или неверно следующее утверждение? Все отчеты должны содержать минимум одну страницу.

Неверно. В некоторых отчетах ничего не печатается. Они называются отчетами только для обработки.

Перечислите имена трех Дизайнеров, используемых совместно с отчетами, и опишите их действие.

Дизайнер отчетов (Report Designer) – конструирование модели данных для отчета.

Дизайнер разделов (Sections Designer) – конструирование макета страницы для отчета.

Дизайнер форм вариантов запроса (Request Options Form Designer)- конструирование вкладки Options формы запроса.

Сколько раз раздел Header печатается в отчете для элемента данных, выделенного отступом (предположим, страница PrintOnEvery выключена, а в таблице есть данные)? Что если элемент данных не выделен отступом?

Один раз для каждой записи в элементе данных, под которым он выделен отступом.

Один раз во всем отчете.

Какое свойство элемента данных может использоваться для удаления вкладки элементов данных с формы запроса (предположим, поля фильтра запроса не установлены)?

DataitemTableView. Вы должны выбрать ключ для сортировки элемента данных.

Какие два свойства элемента данных должны быть установлены для вывода на печать промежуточной суммы поля в разделе нижнего колонтитула группы?

GroupTotalFields и TotalFields.

Когда раздел TransHeader печатается для элемента данных?

Когда разделы области данных элемента данных заполняют одну страницу и должны продолжаться на новой странице. Он не выводится на печать, если новая страница появляется благодаря другому элементу данных.

Какое свойство можно установить, чтобы начинать новую страницу каждый раз при считывании записи?

NewPagePerRecord на элементе данных.

Какие два свойства можно использовать, чтобы напечатать нижний колонтитул на всех страницах?

PrintOnEveryPage=Yes и PrintInBottom=Yes для раздела footer.

Дан отчет, содержащий два элемента данных, второй из которых выделен отступом. Какой C/AL-код могли бы вы написать, чтобы он сильнее всего напоминал эту ситуацию?

```
Nested loops.
       While DataItem1.GetRecord Do
          While DataItem2.GetRecord Do
             ;
```

Какое свойство может использоваться для того, чтобы фильтровать элемент данных, выделенный отступом, на основе полей в не выделенном отступом элементе данных, который находится над ним?

DataitemLink

Глава 5

Основные порты данных

Порты данных используются для экспорта данных во внешние текстовые файлы и для импорта данных из этих файлов.

Глава содержит следующие разделы.

- **Основы порта данных**;
- **Конструирование портов данных**;
- **Примеры портов данных**;
- **Обзор главы**.

5.1 Основы порта данных

Порты данных (dataports) – 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.

Порты данных могут быть *динамическими*, т.е. при выполнении порт данных 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 определяется в форме пользовательского запроса или путем программирования.

На следующей диаграмме представлены компоненты объекта порта данных.

Описание порта данных
- Свойства
- Триггеры
- Элементы данных
  - Свойства
  - Триггеры
  - Поля порта данных
    - Свойства
    - Триггеры
- Форма запроса
  - Свойства
  - Триггеры
  - Элементы управления
    - Свойства
    - Триггеры

На приведенной диаграмме показано, что порт данных состоит из ряда различных компонентов. Далее следует краткое описание каждого компонента.

Описание порта данных

Здесь представлено полное описание порта данных: того, как собираются данные, как они форматируются при записи в выходной файл и т.д. Описание порта данных 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.

Поля порта данных

Поле порта данных (dataport field) относится к полю в файле, из которого должны импортироваться данные. Поля порта данных 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 порт данных. Она используется для получения запросов и опций от пользователя порта данных – например, имени внешнего файла.

Свойство

Свойство (property) – это атрибут объекта (порта данных, элемента данных, поля порта данных и тому подобное), характеризующий некоторым образом данный объект, будь то длина или местоположение поля порта данных в строке (при импорте). Свойства устанавливаются в закладке Property Sheet этого объекта.

Триггер

Предопределенные события, которые происходят с портом данных, позволяют системе выполнять определяемую пользователем C/AL-функцию – событие *запускает* функцию. Как видно из диаграммы, сам порт данных, элементы данных, поля порта данных, форма запроса и элементы управления, имеющиеся на форме запроса, содержат триггеры. Триггеры подвергаются исправлениям в Редакторе C/AL-кода.

Логическая структура

Конструирование порта данных включает две основные задачи: конструирование *модели данных* и определение каркаса *внешнего файла*.

Модель данных

Конструируя *элементы данных* , вы строите модель данных. Элемент данных относится к таблице. При экспорте данных каждый элемент данных повторяется для всех записей соответствующей таблицы. Вы можете устанавливать порядок сортировки, ключи и представления таблицы, которые будут использоваться. Вы можете решать, будет ли конкретная запись заноситься во внешний файл.

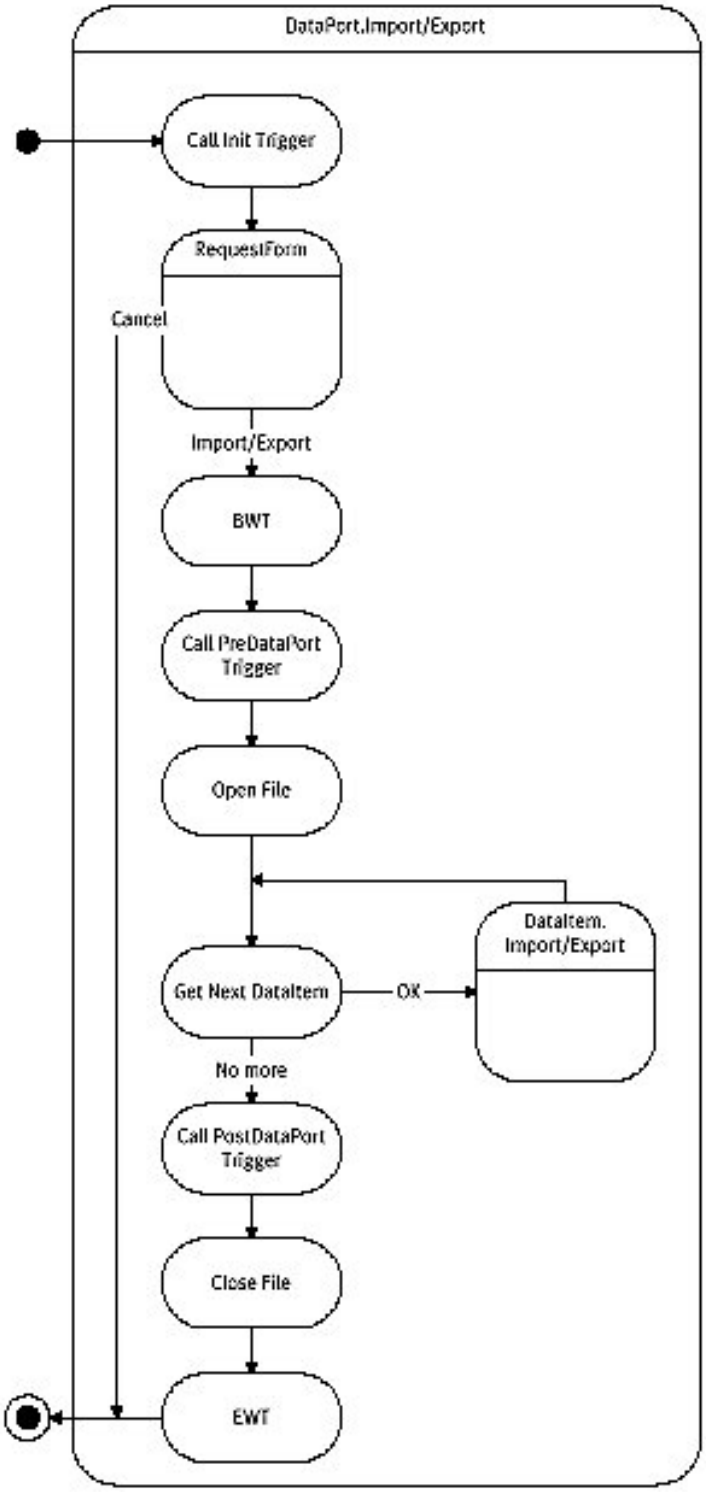
При импорте данных записи, считанные из внешнего файла, могут вставляться в таблицу, относящуюся к данному элементу данных. Вы можете проверять записи перед тем, как вставлять их. Кроме того, можете определять, должны ли записи вставляться автоматически, и должны ли записи, уже находящиеся в БД, быть перезаписаны или обновлены, когда запись с таким же первичным ключом считывается из внешнего файла (и, конечно, должна ли эта запись вообще вставляться).

Внешний файл

Компоновка внешнего файла определяется свойствами порта данных. Во время импорта эти свойства описывают, как входной поток должен разбиваться на записи и поля порта данных; во время экспорта эти свойства описывают, как поля и записи порта данных должны заноситься в файл.

Как запускается порт данных

Следующая блок-схема представляет собой упрощенную версию полного набора блок-схем порта данных в Приложении G, «Блок-схемы порта данных» (*Dataport Flow Charts), в «Руководстве разработчика приложений» (Application Designer's Guide).*



Когда пользователь инициализирует работу порта данных, запускается триггер OnInitDataport. Этот триггер может использоваться для инициализации переменных, но не для вычислений в целом.

При запуске триггера OnInitDataport выполняется форма запроса для порта данных (если она была определена). Здесь пользователь может по своему выбору отменить выполнение порта данных.

Если пользователь выбирает продолжение выполнения, порт данных начинает транзакцию (запускается команда BWT – Begin Write Transaction – начать транзакцию), после чего вызывается триггер OnPreDataport. На этом этапе данные еще не обработаны.

Триггер OnPreDataport может использоваться для обработки данных, введенных пользователем на форме запроса.

Когда триггер OnPreDataport выполнен, открывается внешний файл, и начинается обработка первого элемента данных.

После завершения обработки первого элемента данных следующий элемент данных (если он есть) обрабатывается таким же образом.

Если больше нет элементов данных, вызывается триггер OnPostDataport. Вы можете использовать этот триггер для осуществления любой необходимой пост-обработки.

После завершения работы OnPostDataport внешний файл закрывается.

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 G. Наиболее важный принцип, отображенный на общей схеме, заключается в том, что вся обработка порта данных 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 часто бывает испорчен).

Примечание

Кнопки Right и Left в Дизайнере портов данных (Dataport Designer) позволяют выделить отступом элементы данных в портах данных User Portal. Что касается остальных типов портов данных, то вы должны единовременно обрабатывать по одному элементу данных (таблице).

Сохранение, компиляция и запуск порта данных

После завершения конструирования порта данных вы должны сохранить и откомпилировать его, прежде чем можно будет запускать его из других объектов или из Дизайнера объектов. Обычно это делается после окончания конструирования порта данных. Однако вы можете сохранить порт данных, конструирование которого еще не завершено (из-за чего его нельзя откомпилировать). Кроме того, вы можете откомпилировать и запустить порт данных, не сохраняя и не закрывая его.

Сохранение и закрытие порта данных

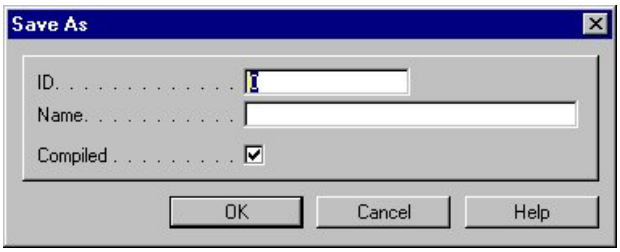
Порт данных закрывается при закрытии окна Дизайнера портов данных. Вы можете закрыть это окно так же, как любое другое.

Чтобы сохранить порт данных, выполните следующие действия.

1 Когда вы закрываете порт данных, среда C/SIDE выведет запрос о том, следует ли сохранить порт данных. Если это новый порт данных (который не сохранялся прежде), вам надо будет присвоить ему идентификатор и имя. Идентификатор должен быть уникальным и соответствовать правилам присваивания номеров объектам – ваш дилер, поставляющий приложения C/SIDE, предоставит вам необходимую информацию.

   Подсказка: если вы введете идентификатор и имя в качестве свойств порта данных, будут использованы эти значения, и вы не получите запрос на ввод идентификатора и имени при закрытии порта данных.

2 Значение поля опции **Compiled** по умолчанию - *TRUE* (отображается как флажок). Если ваш порт данных еще не готов к компиляции, снимите флажок, щелкнув в поле.



3 Нажмите кнопку OK, чтобы сохранить порт данных.

Вы можете сохранить порт данных, не закрывая его, выбрать пункт Save или Save As в меню File. Используя Save As, вы можете

переименовать существующий порт данных (помните, что вы можете скопировать порт данных, открыв и сохранив его под новым именем).

Компиляция порта данных

Порты данных, подобно другим объектам в среде C/SIDE, должны компилироваться до того, как их можно запускать. Как описано выше, вы можете выбрать возможность компиляции порта данных тогда, когда сохраняете его.

Во время конструирования порта данных вы, возможно, захотите произвести его тестовую компиляцию, чтобы найти возможные ошибки. Вы можете произвести тестовую компиляцию порта данных во время конструирования, выбрав опцию Compile в меню Tools.

Запуск порта данных

В финальном приложении ваши порты данных будут включены в меню или будут вызываться, например, кнопкой, расположенной на форме. Однако при конструировании портов данных вам часто будет необходимо запускать их до того, как они будут интегрированы в приложение.

Тестовое выполнение портов данных

Конструируя порт данных, вы можете произвести его тестовую компиляцию, выбрав пункт Run в меню File. В результате порт данных 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 порт данных носит имя Import, фактически записи не будут сохраняться в таблице базы данных во время тестового выполнения.

Запуск портов данных из Дизайнера объектов

Вы можете запускать порт данных из списка портов данных в окне Дизайнера объектов, выбрав его и щелкнув кнопку Run.

5.2 Конструирование портов данных

В данном разделе приводится обзор элементов, из которых состоит конструирование порта данных. В таблице свойств собраны сведения о том, для чего используется каждое свойство. Полные (и самые свежие) сведения даны в онлайновом «Справочном руководстве по C/SIDE» (C/SIDE Reference Guide).

Конструирование порта данных заключается в основном в установке его различных свойств. В последующих разделах даны объяснения того, какие свойства надо использовать и как это делать.

Свойства порта данных

Данный набор свойств определяет порт данных в целом. Кроме того, здесь вы определяете формат внешнего файла.

Обратите внимание, что свойства *Import* и *FileName* могут устанавливаться динамически. Например, вы можете создать порт данных, 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 порта данных).

Свойство	Смысл
ID	Идентификатор порта данных: должен быть уникален, чтобы его можно было отличить от других портов данных.
Name	Имя порта данных: должно быть уникально, чтобы его можно было отличить от других портов данных.
Caption	Заголовок (отображается, например, в окне формы запроса – по умолчанию такой же, как Name).
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 <Yes> позволяет пользователю

Свойство	Смысл
	выбирать импорт или экспорт на вкладке Options формы запроса. Установка значения Yes или No 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 порта данных при вводе или выводе.
FieldEndDelimiter	Когда значение свойства FileFormat – Variable, свойство FieldEndDelimiter используется для определения строки, указывающей на конец поля порта данных при вводе или выводе.
FieldSeparator	Когда значение свойства FileFormat – Variable, свойство FieldSeparator используется для определения строки, разделяющей поля порта данных 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 порт данных.
ShowStatus	Это свойство определяет, будет ли выполняться

Свойство	Смысл
	окно состояния во время функционирования порта данных. Кроме того, данное окно содержит кнопку Cancel, которая позволяет прервать выполнение порта данных – что возможно также в случае, если вы создадите собственное диалоговое окно.
Permissions	Разрешения для порта данных получать доступ к объектам порта данных. (Порт данных может иметь больше полномочий, чем отдельный пользователь, что побуждает пользователя использовать порты данных, извлекающие информацию из таблиц, к которым пользователь обычно не может получить доступ).

FileFormat

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 порта данных. Наконец, свойство *DataItemSeparator* определяет, как должны разделяться элементы данных, если порт данных содержит несколько элементов данных. Обратите внимание, что элементы данных не могут быть вложенными, несмотря на то, что порт данных может содержать несколько элементов данных, которые обрабатываются поочередно.

FileFormat: Fixed

Когда формат внешнего файла – Fixed, поля порта данных в записи имеют фиксированную ширину. Вы можете определить стартовую позицию и ширину каждого поля порта данных в записи (если свойство RecordSeparator – символ *newline,* запись можно представлять себе как строку текста). Позиции и ширина полей порта данных – это свойства полей порта данных (что описано в одном из следующих подразделов).

FileFormat: Variable

Когда формат внешнего файла – Variable, поля порта данных в записи разделены символами, которые вы определяете, и поля порта данных могут иметь различную ширину. Поля порта данных разделяются строкой, определяемой как свойство FieldSeparator.

Свойства *FieldStartDelimiter* и *FieldEndDelimiter* используются для «цитирования» содержания поля порта данных в ситуациях, когда данные поля порта данных содержат символ, который определяется как разделитель (FieldSeparator, RecordSeparator или DataItemSeparator). Наличие этих ограничителей не является обязательным. Это означает, что если должно цитироваться только одно поле порта данных записи, только это поле порта данных должно заключаться в символы FieldStartDelimiter and FieldEndDelimiter. Другие порты данных могут содержать ограничители – на импорт они не будут оказывать влияния. Во время осуществления экспорта все поля порта данных записываются во внешний файл со всеми ограничителями и разделителями.

Свойства элемента данных

Этот набор свойств описывает элементы данных порта данных, являющегося табличной частью БД для порта данных.

Большая часть этих свойств та же и имеет те же функции, что и соответствующие свойства элемента данных в отчете (см. главу «Конструирование отчетов»). Три свойства являются особенностью порта данных: *AutoSave*, *AutoUpdate* и *AutoReplace.*

Свойство	**Смысл**
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 вкладки для данного элемента на форме запроса (по умолчанию – имя DataItemTable).
ReqFilterFields	Имена полей, которые будут первоначально включены в форму ReqFilter.
CalcFields	Имена полей, которые будут вычисляться после извлечения записи.
AutoSave	Это свойство определяет, будут ли импортированные записи автоматически

Свойство	Смысл
	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.

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 and AutoReplace используются для определения того, как записи сохраняются таблице базы данных.

Обратите внимание, что если значение свойства AutoSave - No, 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.

Auto Save (Автосо-хранение)	**Auto Update (Автооб-новление)**	**Auto Replace (Авто-замена)**	**Запись есть в БД и импорти-руемом файле**	**Запись есть только в импортируем ом файле**
Нет*	–-	–-	Запись в БД не обновляется и не заменяется автоматически	Импортируема я запись не вставляется в БД автоматически
Да	Нет	Нет	Произойдет ошибка выполнения, и импорт будет завершен	Импортируема я запись вставляется в БД автоматически
Да	Нет	Да	Импортируема я запись заменит существующую .	Импортируема я запись вставляется в БД автоматически
Да	Да	–-	Импортируема я запись обновит существующую запись в БД	Импортируема я запись вставляется в БД автоматически

–- означает, что настройка не имеет значения

* Если AutoSave – No, можно вставлять или модифицировать запись, используя функцию вставки или модификации из C/AL.

Функция AutoUpdate полезна в некоторых конкретных ситуациях. Ее действие лучше всего объяснить на небольшом примере.

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 вычислены и 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 БД. 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 порта данных

Этот набор свойств определяет поля порта данных записи. Когда значением свойства Format является Fixed, вы используете свойства *StartPos* и *Width* каждого поля порта данных для определения того, как запись, которая считывается из внешнего файла, разбивается на поля порта данных. При экспорте эти свойства определяют, как данные записываются из БД во внешний файл.

Свойство	**Смысл**
Enabled	Определяет, должно ли поле порта данных быть включено или отключено. Отключенное поле порта данных импортируется из внешнего файла, но не будет вставлено в запись.
SourceExpr	Исходное выражение поля порта данных. При импорте им может быть имя поля порта данных таблицы БД, 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 БД, но им может быть любое допустимое выражение C/AL
StartPos	Если значением свойства FileFormat является Fixed, это – стартовая позиция данного поля порта

Свойство	Смысл
	данных. Нумерация позиций начинается с 1.
Width	Если значением свойства FileFormat является Fixed, это – ширина данного поля порта данных.
CallFieldValidate	Определяет, будет ли выполняться триггер OnValidate при импорте поля порта данных.
Format	Описывает, как поле порта данных будет форматироваться во время экспорта. Например, вы можете определить количество десятичных разрядов и т.д.

Перед выполнением экспорта поля данные конвертируются в текст. Если формат имеет значение Fixed, а ширина (Width) меньше, чем фактическая ширина строки данных после конвертации, содержимое будет усечено справа так, чтобы оно имело определенную ширину. То есть номер округляется и усекается не как номер, а как текст – справа.

Вы получите предупреждение во время конструирования, если вы определили поля порта данных с такими стартовыми позициями и шириной, которые приведут к перекрытию этих полей портов данных. Ошибка делает невозможной компиляцию (и, соответственно, выполнение) порта данных.

Однако вы можете иметь разрывы в полях порта данных (стартовая позиция второго поля порта данных не обязательно должна быть там, где закончилось первое поле порта данных). Благодаря этим разрывам можно пропускать ненужную информацию или создавать пустые столбцы в файле.

Во время импорта значение, которое слишком велико для конкретного типа данных или заданной ширины поля таблицы БД, куда оно должно быть вставлено, вызовет ошибку, и выполнение прекратится. Поскольку весь порт данных находится в транзакции, аварийное завершение не испортит БД.

5.3 Примеры портов данных

В данном разделе рассказано о том, как создавать порты данных, последовательно производя действия, необходимые для создания четырех основных типов портов данных - осуществлять импорт и экспорт как с фиксированным, так и с переменным форматом внешнего файла.

Упражнение 5.3.1 Экспорт – фиксированный формат

В этом примере порт данных будет экспортировать записи в файл в фиксированном формате. Записи в файле будут разделяться новыми строками, и в каждой записи (т.е. строке) поле порта данных будет иметь одинаковую ширину во всех записях, вне зависимости от фактической ширины строки данных поля порта данных.

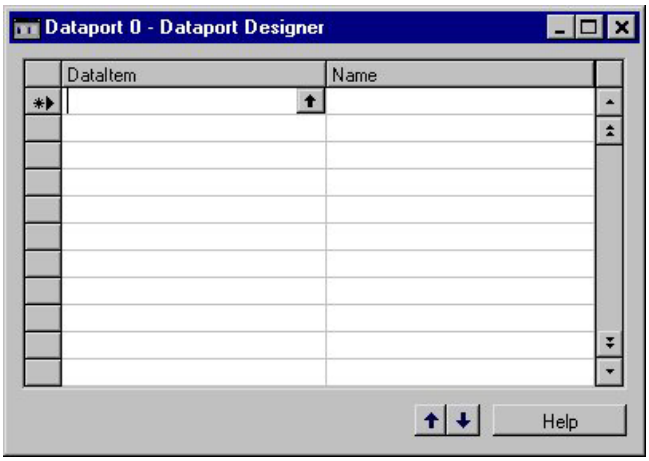
В качестве примера таблицы БД будет использоваться **G/L Account** (Chart of Accounts – схема счетов). Существует несколько полей FlowFields среди экспортируемых полей портов данных, которые должны вычисляться при экспорте.

Простая версия

Первая версия – очень простая. В следующем подразделе она будет усовершенствована.

1 Откройте Дизайнер объектов.

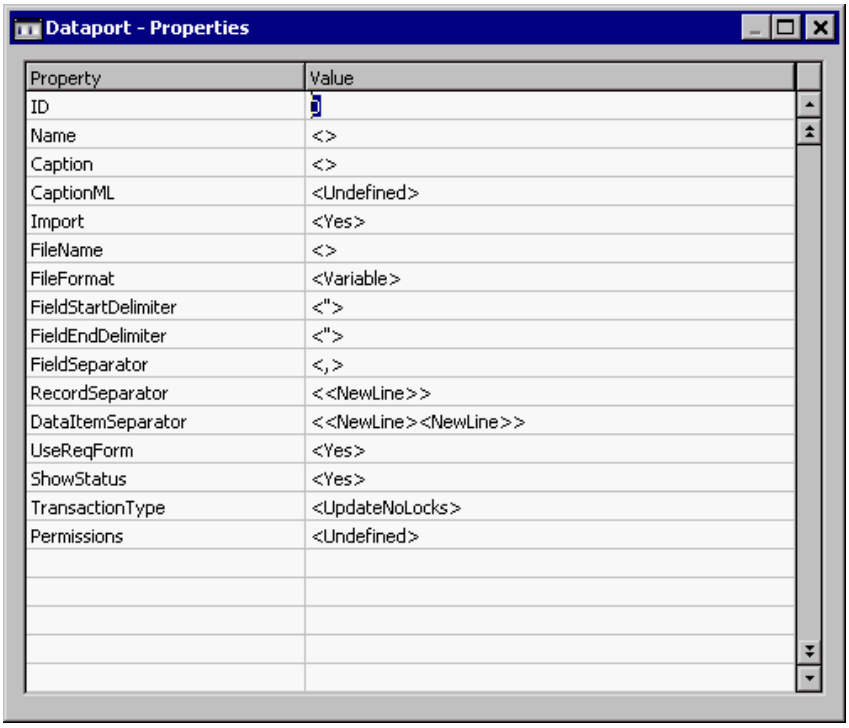
2 Нажмите Dataport, New.



3 Приложение C/SIDE открывает Дизайнер портов данных. Выберите поле **DataItem** и таблицу **G/L Account** в форме поиска, которую вы открываете, нажав кнопку Lookup. Значение Name

установлено по умолчанию как имя таблицы. В этом порту данных менять его не нужно.

4 Откройте Property Sheet для порта данных (не для элемента данных). Для этого нажмите кнопку View, Properties, пока порт данных выбран (вы можете выбрать порт данных, либо щелкнув в пустой строке в Дизайнер портов данных, либо используя Edit, Select Object).



5 Установите следующие свойства:

Import - задайте значение *No*, чтобы создать порт данных, экспортирующий данные.

FileFormat – задайте значение *Fixed*.

Значения остальных свойств можно оставить по умолчанию.

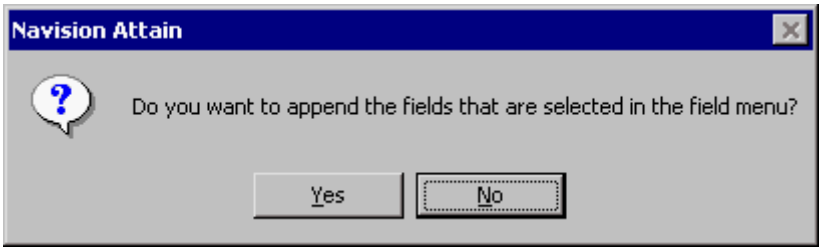
6 Выберите элемент данных G/L Account, щелкнув его в окне Дизайнера портов данных.

7 Щелкните View, Dataport Fields, и откроется Dataport Field Designer.

8 В окне Dataport Field Designer щелкните кнопку View, Field Menu, и появится окно **Field Menu**.

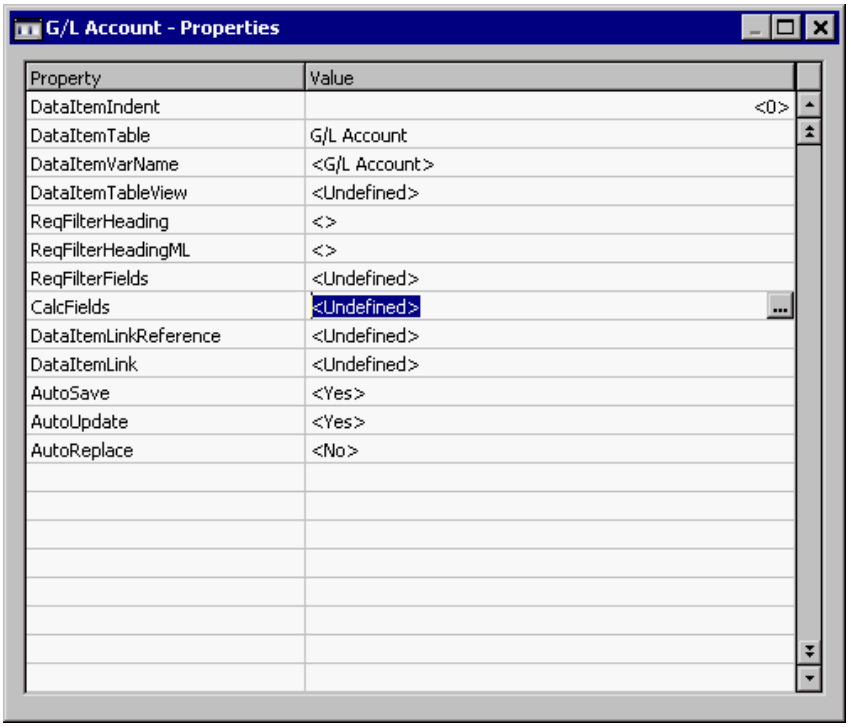
9 Выберите поля, которые необходимо экспортировать. Здесь выбраны поля **No.**, **Name**, **Balance at Date** и **Net Change**. Возможно, вам придется прокрутить вниз окно **Field Menu,** чтобы выбрать эти поля, в зависимости от установленного размера окна **Field Menu**.

10 Щелкните в окне Dataport Field Designer, чтобы добавить поля, которые вы только что выбрали в окне **Field Menu**. (Вы не можете нажать кнопку OK в окне **Field Menu**). Вы получите запрос о том, хотите ли вы прикрепить выбранные поля.

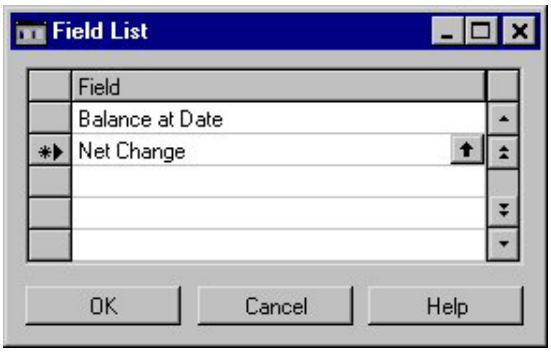


11 Нажмите кнопку Yes.

12 Перейдите в Property Sheet элемента данных G/L Account.



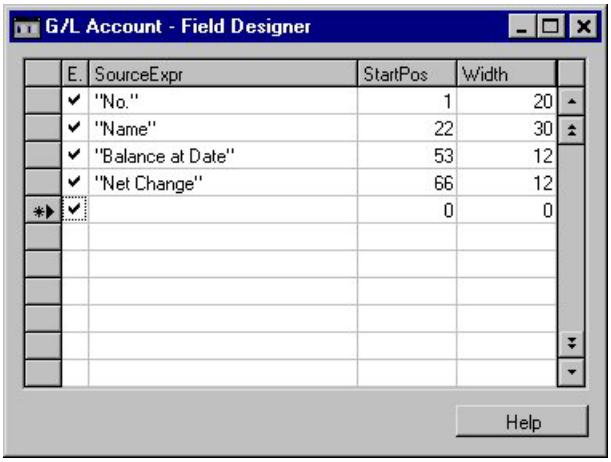
Выберите свойство CalcFields, щелкните AssistButton, чтобы выбрать окно **Field List**. Затем выберите те поля из элемента данных, которые должны вычисляться при экспорте, то есть FlowFields. В этом случае это поля **Balance at Date** и **Net Change**. Нажмите OK, чтобы закрыть окно **Field List**.



Теперь вы готовы запустить порт данных. Вы можете запустить порт данных до его сохранения, нажав кнопку File, Run. Если вы сначала хотите сохранить порт данных, сначала вам надо будет задать имя и номер, как и при работе с другими объектами.

О вставке полей порта данных

После того как поля порта данных были вставлены из Field Menu, вы можете проверить установку свойств StartPos и Width всех полей в форме Dataport Field Designer (к которой можно получить доступ, нажав кнопку View, Dataport Fields). Они выглядят следующим образом:



Как видите, оба свойства – StartPos и Width – за вас уже заполнены. Им присваиваются значения согласно следующим правилам.

Тип данных поля	Присвоенная ширина
Code	Если фактически длина > 10, используется фактическая длина, иначе - 10
Text	Если фактически длина > 30, используется фактическая длина, иначе - 30
Date	11
Time	10
Option	10
Decimal	12
Integer	7
Boolean	10

Выполнение порта данных

Запустив порт данных, вы увидите форму запроса, заданную по умолчанию. Первая вкладка не представляет особого интереса, но вторая, Options, важна. Здесь вы можете задать имя внешнего файла, в который будет осуществляться запись.

Указав имя файла, нажмите кнопку OK, чтобы запустить порт данных. Окно состояния, заданное по умолчанию, покажет ход выполнения. После завершения задачи вы можете просмотреть файл в текстовом редакторе.

d:\a\FIN\W1\test.txt

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	1,479,522.74	1,479,522.74
1120	Increases during the Year	147.74	147.74
1130	Decreases during the Year	0	0
1140	Accum. Depreciation, Buildings	-526,635.37	-526,635.37
1190	Land and Buildings, Total	953,035.11	953,035.11
1200	Operating Equipment	0	0
1210	Operating Equipment	582,888.78	582,888.78
1220	Increases during the Year	0	0
1230	Decreases during the Year	0	0
1240	Accum. Depr., Oper. Equip.	-505,089.12	-505,089.12
1290	Operating Equipment, Total	77,799.66	77,799.66
1300	Vehicles	0	0
1310	Vehicles	49,475.32	49,475.32
1320	Increases during the Year	0	0
1330	Decreases during the Year	0	0
1340	Accum. Depreciation, Vehicles	-47,255.12	-47,255.12
1390	Vehicles, Total	2,220.2	2,220.2
1395	Tangible Fixed Assets, Total	1,033,054.97	1,033,054.97

Улучшенная версия

Этот простой порт данных можно сделать более удобным для пользователя. В зависимости от того, для выполнения каких задач будет использоваться порт данных, можно изменить несколько параметров. Ниже даны примеры того, что вы можете сделать:

- первая вкладка в форме запроса не должна быть видна, т.к. мы не хотим, чтобы пользователь устанавливал фильтры и ключи;
- экспортироваться должны только те счета, в которых свойство Account Type имеет значение Posting или End-Total;
- числа должны иметь формат тысяч. Не должно быть разделителей групп разрядов, десятичных чисел, а знак перед числом должен присутствовать.

Изменение формы запроса

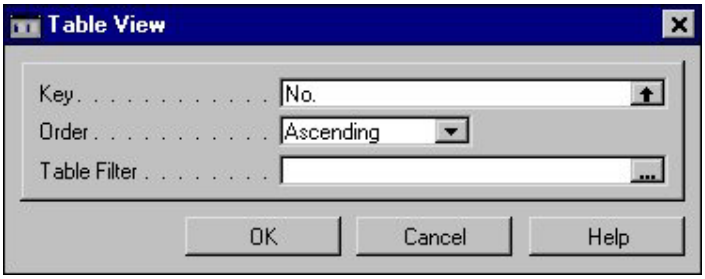
Нельзя устанавливать значение *No* в свойстве UseReqForm, чтобы удалить форму запроса. Такое действие приведет к тому, что пользователь не сможет задавать имя файла, в который должны записываться данные. Фактически, поскольку имя файла не будет задано, порт данных вызовет ошибку выполнения.

Вместо этого вы можете выбрать вкладку **Options** и удалить вкладку элемента данных, установив DataItemTableView для элемента данных. После установки этого свойства – в противоположность тому, что оно остается не заданным – пользователь не может изменять ключ или

порядок сортировки и не может задавать фильтр таблицы, а соответствующая вкладка будет удалена из системы.

1 Откройте Property Sheet для элемента данных.

2 Щелкните кнопку AssistButton в свойстве DataItemTableView, чтобы открыть эту форму.

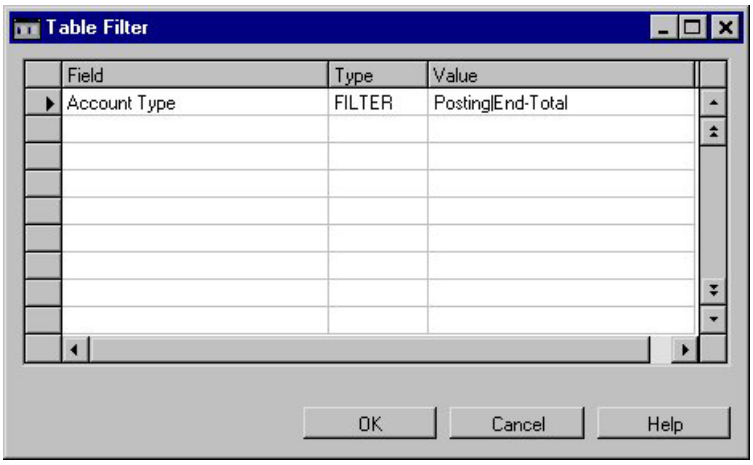


3 Заполните хотя бы одно поле. На предыдущем рисунке для **Key** установлено значение *No.*, а для **Order** – *Ascending.*

Выбор определенных типов учетных записей

Чтобы выбрать только некоторые типы учетных записей, вы должны установить фильтр таблицы. Это можно сделать в свойстве DataItemTableView. Продолжайте выполнять описанные выше действия, начиная с пункта 3.

4 Нажмите кнопку AssistButton в поле Table Filter, чтобы открыть следующую форму.



5 Установите значения полей **Field**, **Type** и **Value**, как показано на предыдущем рисунке. В результате будет создан фильтр

таблицы, который выбирает записи, где значение *Account Type* определено как *Posting or End-Total* (символ, стоящий между этими двумя значениями, - | (вертикальная черта), означающий ИЛИ).

6 Нажмите кнопку OK, чтобы закрыть форму **Table Filter** и форму **Table View**.

Изменение формата чисел

Для экспорта десятичных полей **Balance at Date** и **Net Change** в качестве тысяч (так что число 1.444.723,67 экспортируется как `1444)` вы должны использовать свойство SourceExpr следующих полей.

7 Перейдите в Property Sheet полей порта данных **Balance at Date** и **Net Change**.

8 Введите следующее выражение как свойство SourceExpr первого поля порта данных:

```
FORMAT(ROUND("Balance at Date"/1000,1,'='),0,1)
```

В C/AL-выражении поле **Balance at Date** сначала делится на 1000. Результат округляется функцией ROUND, а затем функция FORMAT используется для передачи возвращаемого значения из функции ROUND в формате 1 (которое в случае десятичного значения имеет вид <Sign><Integer><Decimals>).

9 Введите следующее выражение как свойство SourceExpr второго поля порта данных:

```
FORMAT(ROUND("Net Change"/1000,1,'='),0,1)
```

Файл, созданный этим портом данных, может выглядеть так при импорте в электронную таблицу:

test2.txt

	A	B	C	D	E
1	1110	Land and Buildings	1480	1480	
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	0	0	
8	1230	Decreases during the Year	0	0	
9	1240	Accum. Depr., Oper. Equip.	-505	-505	
10	1290	Operating Equipment, Total	78	78	
11	1310	Vehicles	49	49	

test2

Упражнение 5.3.2 Экспорт – переменный формат

В этом примере порт данных будет экспортировать те же записи, что и в прошлом примере, но в переменном формате. Каждое поле порта данных в записи будет отделено символами, которые вы определите, и будет иметь фактическую ширину данных поля порта данных в каждой записи.

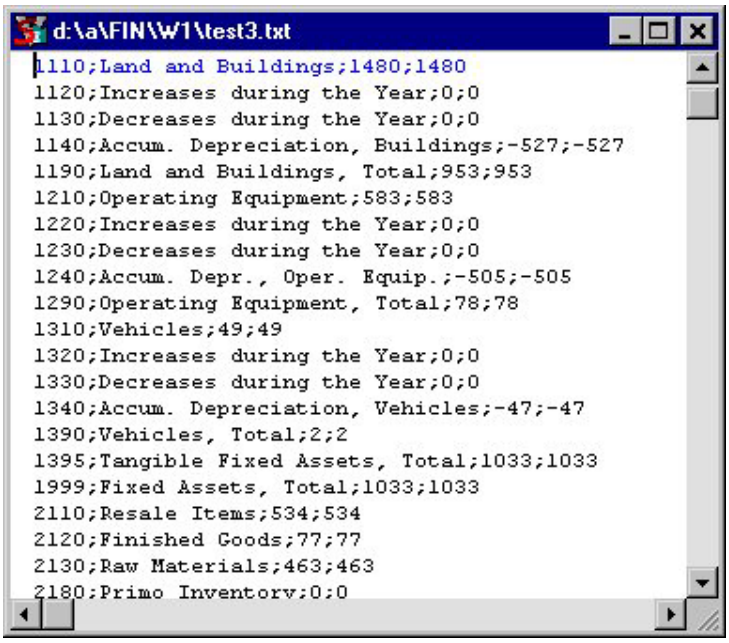
Для создания такого порта данных вместо порта данных с фиксированным форматом ширины необходимо внести совсем немного изменений. Вы должны определить нужный набор опций для элемента данных, как показано ниже.

Dataport - Properties

Property	Value
ID	0
Name	<>
Caption	<>
CaptionML	<Undefined>
Import	No
FileName	<>
FileFormat	Fixed
FieldStartDelimiter	<">
FieldEndDelimiter	<">
FieldSeparator	<;>
RecordSeparator	<<NewLine>>
DataItemSeparator	<<NewLine><NewLine>>
UseReqForm	<Yes>
ShowStatus	<Yes>
TransactionType	<UpdateNoLocks>
Permissions	<Undefined>

Свойство FileFormat получает значение *Variable,* а свойство FieldSeparator – значение ; (точка с запятой). Свойства FieldStartDelimiter и FieldEndDelimiter устанавливаются на *<None>.* Значения свойств RecordSeparator и DataItemSeparator остаются по умолчанию, что означает, что записи будут отделены новыми строчками, а элементы данных – двумя новыми строчками (что в действительности не представляет интереса здесь, т.к. у нас есть только один элемент данных в этом порту данных).

Экспортируемый файл может выглядеть так, как показано ниже в текстовом редакторе.



Точка с запятой использовалась как разделитель FieldSeparator, потому что поля порта данных содержат и символы пробела, и запятые. Другим решением могло бы быть использование ограничителей. В этом случае, разделителем FieldSeparator могла бы также быть запятая, но ваш выбор должен зависеть от того, какое приложение будет использоваться для открытия экспортируемого файла и какие форматы это приложение поддерживает при импорте текстовых файлов.

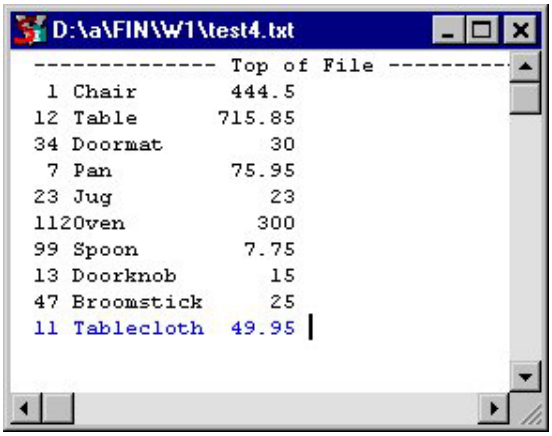
Упражнение 5.3.3 Импорт – фиксированный формат

Создание порта данных, поддерживающего импорт данных, не очень отличается от создания порта, экспортирующего данные. Тем не менее, вы должны учитывать то, как импортируемые записи должны вставляться в таблицу БД, лежащую в основе элемента данных. Это

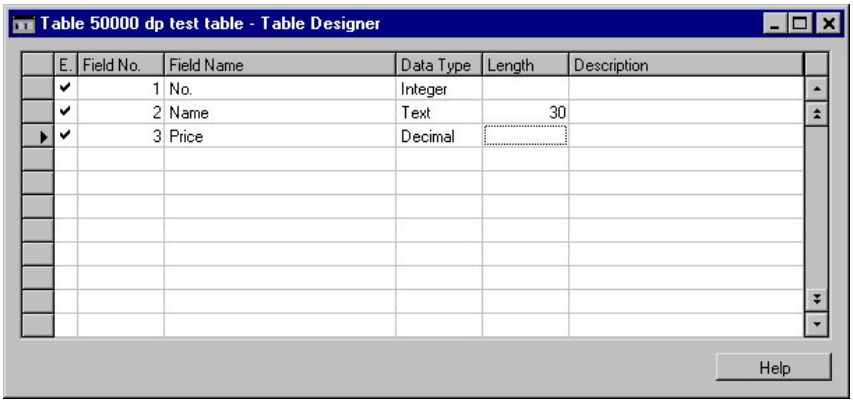
особенно важно, если таблица уже содержит записи с таким же первичным ключом, как у некоторых записей, подлежащих импортированию.

В первом примере подразумевается, что таблица пуста (или что были предприняты другие меры для предотвращения конфликтов).

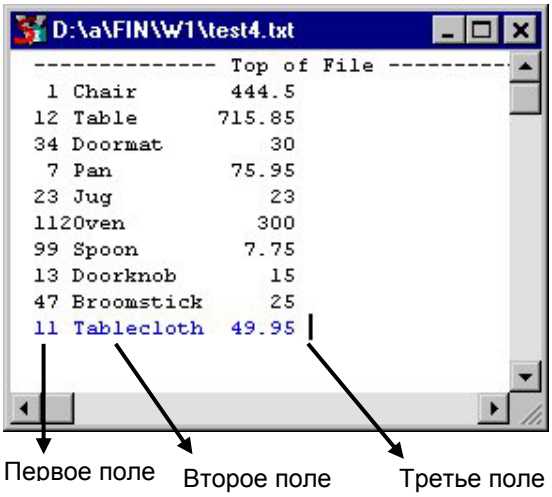
Файл, из которого мы будем импортировать данные, выглядит следующим образом в текстовом редакторе.



В данном случае мы будем импортировать записи в новую таблицу. Таблица имеет следующую компоновку.



Это означает, что мы должны решить, как строки в импортируемом файле (каждая строка станет записью в элементе данных) будут разбиваться на поля порта данных. Строки имеют фиксированный формат, и внимательно изучив компоновку строк, можно определить стартовые позиции этих полей, отображенные в нижней строке.



Она подсказывает нам, что поля порта данных должны содержать следующие свойства:

Поля порта данных	**Начальная позиция (StartPos)**	**Ширина (Width)**
No.	1	3
Name	4	11
Price	15	6

После определения того, как организованы строки в импортируемом файле, создание порта данных становится простой задачей.

1 Создайте порт данных так, как описано ранее.

2 Установите значение *Yes* в свойстве Import и значение *Fixed* – в свойстве FileFormat.

3 В этом примере используется предыдущая таблица. Для этой таблицы должен быть создан элемент данных.

4 Добавьте все поля из таблицы в этот порт данных, используя Dataport Field Designer и Field Menu, как описано выше. На этом этапе будут заданы поля портов данных такими с размерами (и соответствующими стартовыми позициями), как определено благодаря изучению компоновки таблицы.

Он будет выглядеть следующим образом:

dp test table - Field Designer

E.	SourceExpr	StartPos	Width
✔	"No."	1	7
✔	"Name"	9	30
✔	"Price"	40	12
✔		0	0

Help

5 Вы должны установить такие значения StartPos и Width, которые подходят конкретному файлу, откуда вы собираетесь импортировать данные. В данном примере значения нужно установить так, как показано ниже (это описано ранее):

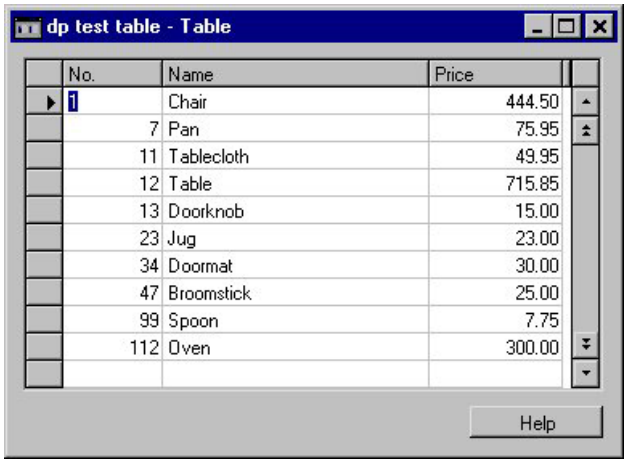
dp test table - Field Designer

E.	SourceExpr	StartPos	Width
✔	"No."	0	0
✔	Price	0	0
✔	Name	0	0

Help

6 Порт данных можно запускать. Помните, что если вы запустите его из окна Дизайнера портов данных (нажав кнопку File, Run), записи не будут сохранены в БД (чтобы сохранить их, вы должны запустить порт данных из Дизайнера объектов или вызвать его из меню). Кроме того, вы можете удалить ненужные вкладки из формы запроса.

После выполнения порта данных, благодаря чему записи фактически импортированы в БД, таблица будет выглядеть следующим образом:



No.	Name	Price
1	Chair	444.50
7	Pan	75.95
11	Tablecloth	49.95
12	Table	715.85
13	Doorknob	15.00
23	Jug	23.00
34	Doormat	30.00
47	Broomstick	25.00
99	Spoon	7.75
112	Oven	300.00

Обратите внимание, что, поскольку Field No. является первичным ключом таблицы, записи отображаются в порядке, определенном этим полем – который, кстати, не такой, в каком они располагались в импортируемом файле.

Возможные ошибки

Легко допустить ошибки при решении вопроса о том, как отсекать строки в импортируемом файле. В некоторых случаях они приводят к ошибкам периода выполнения при запуске порта данных. Например, представьте себе, что мы допустили ошибку при установке поля порта данных **Field No.**, в результате чего ему присвоена ширина, которая превышает фактическую на один символ. Для большей части импортируемого файла это не имеет значения – оставшееся место будет игнорироваться. Но строка, начинающаяся с «112Oven..."», вызовет ошибку периода выполнения, когда C/SIDE прочтет 112O вместо 112. «O» (верхний регистр от «o») нельзя вставить в поле для целых чисел.

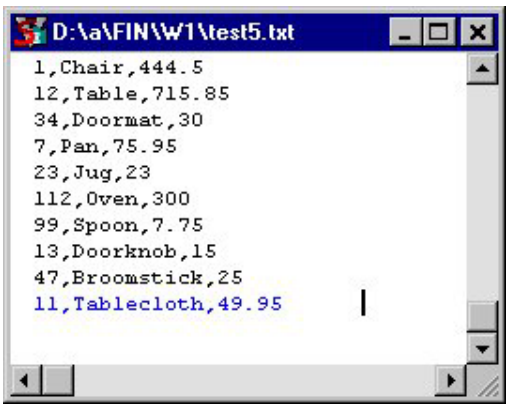
Можно считать удачей, если ошибка вызывает ошибку периода выполнения. В других случаях ошибка может остаться незамеченной C/SIDE, например, если «разрез» между двумя полями был помещен некорректно. Если есть возможность, тщательно протестируйте импортированные данные перед их использованием в работе.

Подсказка

В некоторых случаях порядок полей, которые вам нужны в таблице, не соответствует порядку полей порта данных в импортируемом файле. В таком случае вы можете просто сконструировать поля порта данных в том порядке, который вам нужен, добавляя конкретные поля. Будет необходимо установить свойства StartPos и Width вручную.

Упражнение 5.3.4 Импорт – переменный формат

Изменение импортирующего порта данных для работы с внешним файлом в переменном формате – несложная задача. Предположим, файл выглядит следующим образом.



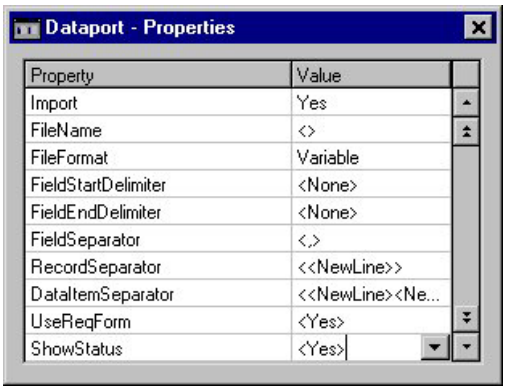
Данные точно такие же, как в файле с фиксированным форматом, из которого мы импортировали ранее, но здесь поля порта данных разделены запятыми. Единственные различия в создании портов данных заключаются в следующем.

1 Установите значение свойства FileFormat на *Variable*

2 Установите значение FieldSeparator на запятую.

3 Установите ограничители на <None>.

Свойства порта данных теперь должны выглядеть так:

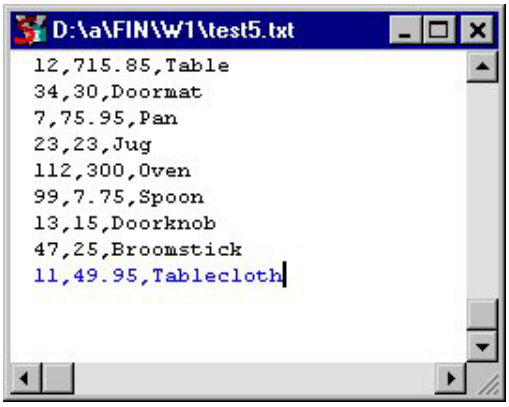


В этом все различия между портом данных с фиксированным форматом и портом данных с переменным форматом. Результат выполнения этого порта данных будет таким же, как результат выполнения порта данных с фиксированным форматом.

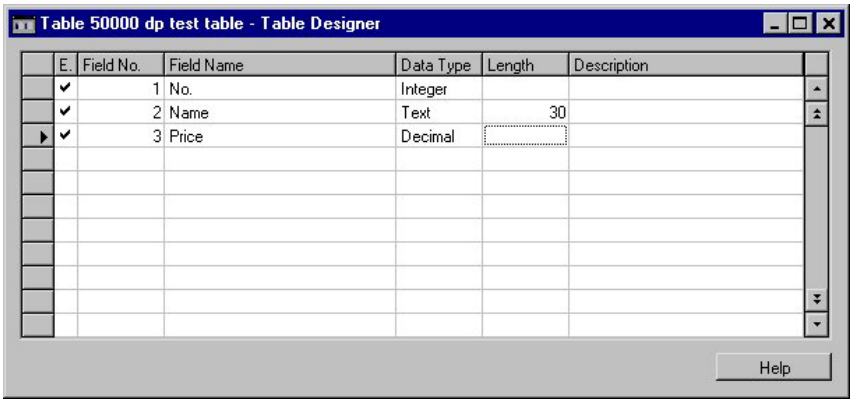
Примечание

Если порядок полей порта данных в импортируемом файле отличается от порядка полей порта данных в элементе данных, вы не можете использовать тот же метод для перемещения полей порта данных, какой вы применяли бы для файла с фиксированным форматом. Вместо этого вы можете изменить порядок полей порта данных в Dataport Field Designer.

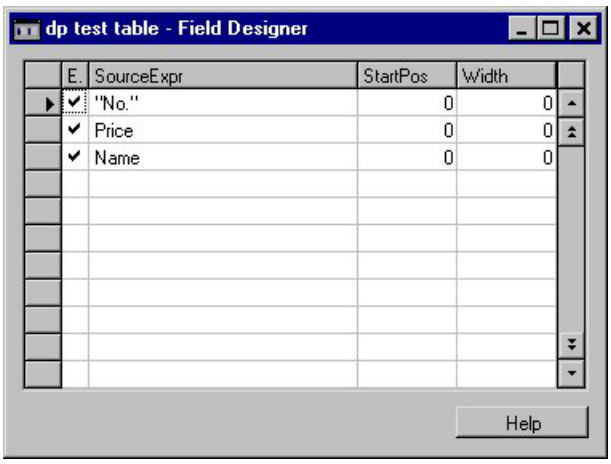
Предположим, импортируемый файл имел следующий формат:



Допустим, у таблицы следующая компоновка (такая же, как описана выше):



В данном случае вы можете добавить поля порта данных в Dataport Field Designer в том порядке, в каком они представлены в импортируемом файле:



Обратите внимание, что, поскольку поля порта данных были добавлены по одному из Field Menu (а не все сразу), значение свойств **StartPos** и **Width** не было вычислено Дизайнером. Однако это не имеет значения, потому что эти свойства не используются в порту данных с переменным форматом.

5.4 Обзор главы

В этом разделе содержатся вопросы, касающиеся материала данной главы. Ответы на вопросы можно найти в подразделе «Ответы».

Вопросы

1 В чем различие между описанием отчета и описанием порта данных?

2 Что можно сделать с элементом данных в порту данных такого, что применимо и к элементу данных в отчете?

3 Чему должны соответствовать поля порта данных, которые вы создаете в Dataport Field Designer, для импортирующего порта данных?

4 Что импортирующий порт данных будет делать с записями из файла, если вы установите значение свойства AutoSave на *No,* а AutoUpdate - на *Yes*?

5 Когда необходимо использовать свойства StartPos и Width поля порта данных?

6 Верно или неверно следующее утверждение? Каждое поле порта данных должно быть связано с реальным полем в элементе данных.

7 Что свойство Calcfields делает с элементом данных?

8 Какой тип файла порта данных выравнивает данные в столбцах при экспорте?

9 Какой тип файла порта данных создает наименьший файл при экспорте?

10 Какие свойства порта данных вы можете установить так, что пользователь не будет видеть вкладку с опциями при импорте?

Ответы

1 В чем различие между описанием отчета и описанием порта данных?

Порт данных содержит поля порта данных, а не разделы.

2 Что можно сделать с элементом данных в порту данных такого, что применимо и к элементу данных в отчете?

Выделить отступом элементы данных.

3 Чему должны соответствовать поля порта данных, которые вы создаете в Dataport Field Designer, для импортирующего порта данных?

Файлу, который вы хотите импортировать.

4 Что импортирующий порт данных будет делать с записями из файла, если вы установите значение свойства AutoSave на *No,* а AutoUpdate - на *Yes*?

Ничего. Если свойство AutoSave установлено на No, порт данных не будет вставлять или модифицировать какие-либо записи автоматически.

5 Когда необходимо использовать свойства StartPos и Width поля порта данных?

Если тип файла – Fixed.

6 Верно или неверно следующее утверждение? Каждое поле порта данных должно быть связано с реальным полем в элементе данных.

Неверно. Поля порта данных могут быть связаны с переменными или другими полями в переменных записей (при экспорте другие значения тоже приемлемы).

7 Что свойство Calcfields делает с элементом данных?

Оно инструктирует порт данных вычислить значения FlowFields, которые должны экспортироваться.

8 Какой тип файла порта данных выравнивает данные в столбцах при экспорте?

Fixed.

9 Какой тип файла порта данных создает наименьший файл при экспорте?

Переменный. Между полями порта данных нет дополнительного пространства.

10 Какие свойства порта данных вы можете установить так, что пользователь не будет видеть вкладку с опциями при импорте?

Свойство Filename – на имя файла (пользователь не сможет изменить его). Свойство Import – на Yes (пользователь не сможет изменить его). Вкладка с опциями исчезает.

Глава 6

Основные программные модули

В этой главе дается объяснение того, что такое программный модуль и как его создавать. В ней также показано, как в программном модуле использовать функции из других прикладных объектов.

Глава содержит следующие разделы.

- **Программный модуль среды C/SIDE**;
- **Создание программных модулей**;
- **Использование программных модулей**;
- **Обзор главы**.

6.1 Программный модуль среды C/SIDE

Возможно, вы уже встречали примеры программного C/AL-кода в 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

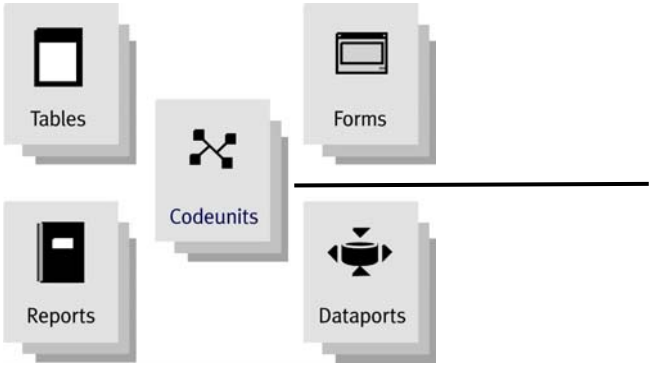
Функция (function) – это последовательность выражений на языке C/AL, которую вы определяете с целью создания новых функциональных возможностей. В каждой функции можно определить локальные переменные, то есть переменные, область действия которых ограничена той функцией, где они определены.

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.



В программном модуле вы можете хранить общие функции, не связанные напрямую с каким-либо из ваших прикладных объектов

Каждая функция, которую вы добавляете к программному модулю, будет отображаться в отдельной секции при просмотре файла в Редакторе C/AL-кода.

```
Codeunit 2 Company-Initialize - C/AL Editor
Documentation()

OnRun()
Window.OPEN('Initializing company...');

AccountingSetup.INIT;
AccountingSetup.INSERT;

IF NOT (SourceCodeSetup.FIND('-') OR SourceCode.FIND('-
  WITH SourceCodeSetup DO BEGIN
    INIT;
    InsertSourceCode(Sales,'SALES','Sales');
    InsertSourceCode(Purchases,'PURCHASES','Purchases')
    InsertSourceCode("Inventory Post Cost",'INVTPCOST',
    InsertSourceCode("Exchange Rate Adjmt.",'EXCHRATADJ
    InsertSourceCode("Post Recognition",'POSTRECOG',Rep
    InsertSourceCode("Post Value",'POSTVALUE',ReportNam
    InsertSourceCode("Close Income Statement",'CLSINCOM
    InsertSourceCode(Consolidation,'CONSOLID','Consolid
    InsertSourceCode("General Journal",'GENJNL',FormNam
    InsertSourceCode("Sales Journal", 'SALESJNL',FormNa
```

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

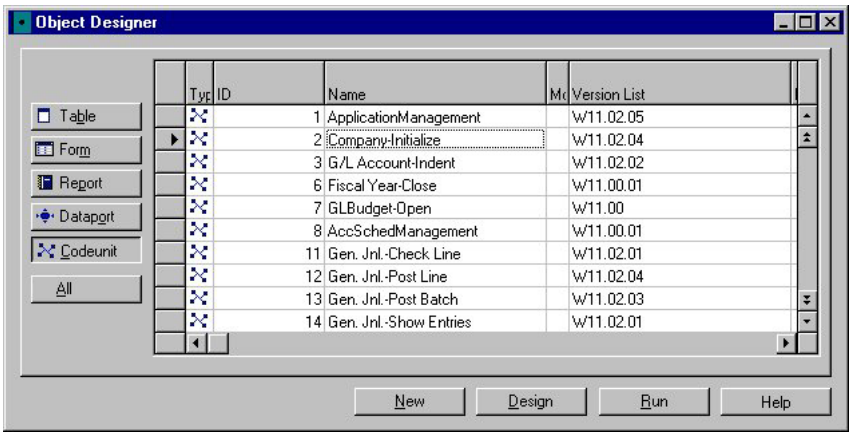
Программные модули содержат функции, но тоже могут исполняться. Будучи контейнером для функций, которые можно запускать по отдельности, программный модуль сам может исполняться при условии написания кода `<CodeunitVariableName>.Run`. Запуская программный модуль, вы исполняете код в секции OnRun программного модуля.

6.2 Создание программных модулей

Вы можете создавать новый программный модуль или модифицировать существующий теми же средствами, что другие прикладные объекты, то есть с помощью Дизайнера объектов (Object Designer).

Чтобы создать программный модуль, выполните следующие действия.

1 Чтобы открыть Дизайнер Объектов, нажмите Tools, Object Designer.



2 Нажмите в Дизайнере объектов кнопку Codeunit.

3 Чтобы создать новый программный модуль, нажмите кнопку New. Среда C/SIDE откроет Редактор C/AL-кода, в котором вы можете создавать функции.

Чтобы модифицировать существующий программный модуль, выполните следующие действия.

1 Нажмите кнопку Codeunit в Дизайнере объектов.

2 Выберите программный модуль, который вы хотите модифицировать.

3 Нажмите кнопку Design, и среда C/SIDE откроет Редактор C/AL-кода. В нем вы сможете модифицировать программный модуль, изменяя существующие или добавляя новые функции.

Использование Редактора C/AL-кода

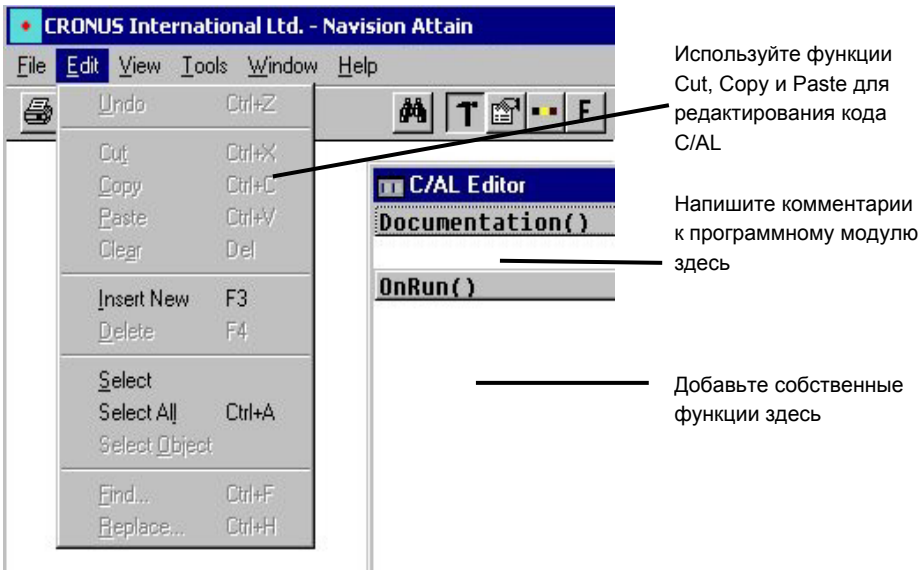
Редактор C/AL-кода (C/AL Editor) служит для просмотра и редактирования программного кода. Этот редактор предназначен для облегчения задачи создания и модификации C/AL-кода. Открыв его, вы получаете доступ к меню C/AL Symbol Menu, которое помогает вам определять функции C/AL. Используя C/AL Symbol Menu, вы можете

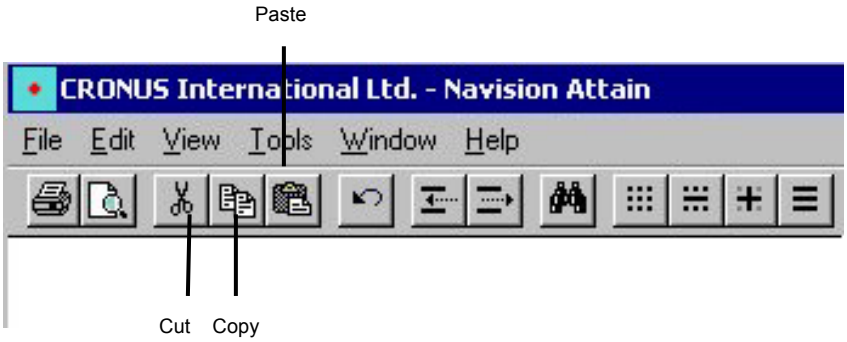
получить справку обо всех функциях C/AL. Выберите имя функции C/AL в столбце справа и нажмите клавишу F1.

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 Edit, так и из панели инструментов.





При работе в Редакторе C/AL-кода вы можете использовать несколько сочетаний клавиш.

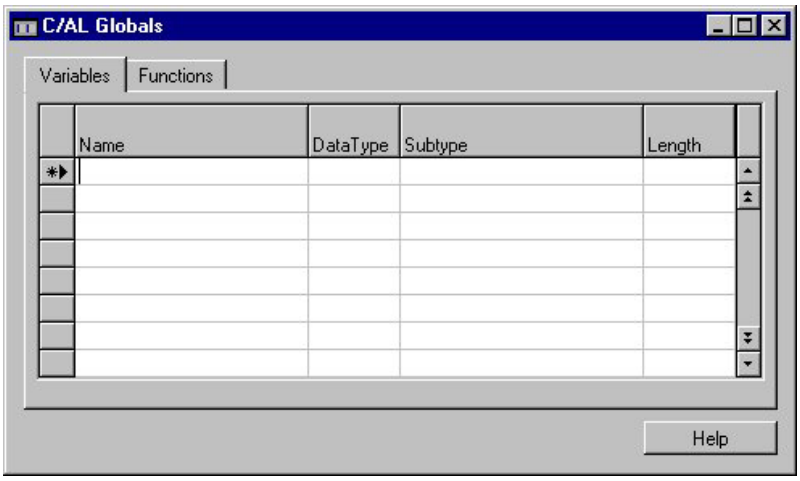
Чтобы…	Нажмите…
перенести выбранный текст в буфер обмена	CTRL+X
скопировать выбранный текст в буфер обмена	CTRL+C
вставить текст из буфера обмена в программный модуль в точку нахождения курсора	CTRL+V

Определение переменных и функций в программных модулях

Создав программный модуль, вы должны сделать следующий шаг – определить глобальные переменные и функции для этого программного модуля. В этом случае используется такой инструмент, как C/AL Globals.

Чтобы получить доступ к инструменту C/AL Globals, выполните следующие действия.

1 Убедитесь в том, что выбран Редактор C/AL-кода. Нажмите кнопку View, затем C/AL Globals, и приложение C/SIDE отобразит окно **C/AL Globals**.



2 В окне **C/AL Globals** выберите закладку либо для добавления глобальной переменной (Variables), либо для добавления функции (Functions).

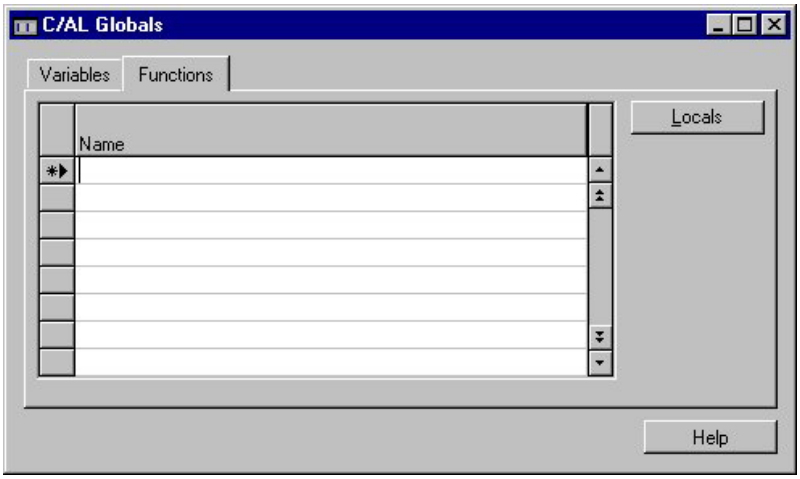
Чтобы добавить глобальную переменную, выполните следующие действия.

1 Нажмите закладку **Variables** в окне **C/AL Globals**.

Добавьте имя и тип. 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). Если вы выбираете OCX или Automation, необходимо также добавить подтип.

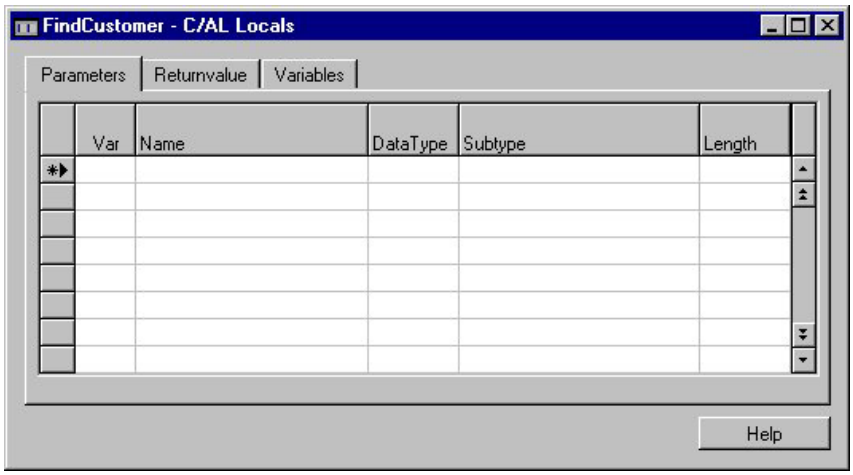
Чтобы добавить функцию, выполните следующие действия.

1 Нажмите закладку Functions в окне **C/AL Globals**. Среда C/SIDE отобразит следующее окно.



2 Введите имя для каждой функции, которую вы хотите добавить.

3 Нажмите кнопку Locals, чтобы определить параметры, возвращаемое значение и локальные переменные для каждой функции. Приложение C/SIDE отобразит окно **C/AL Locals**.

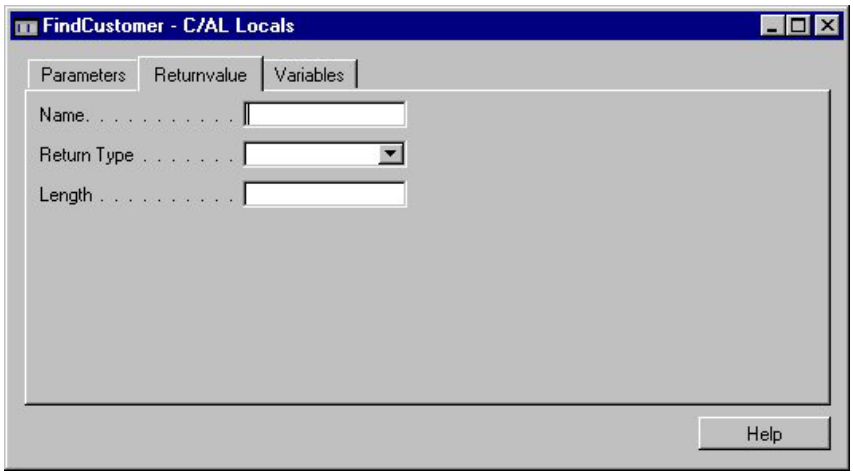


4 Для каждого параметра вы должны указать метод вызова, имя и тип данных. Вы можете определить подтип и длину, но это необязательно.

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.

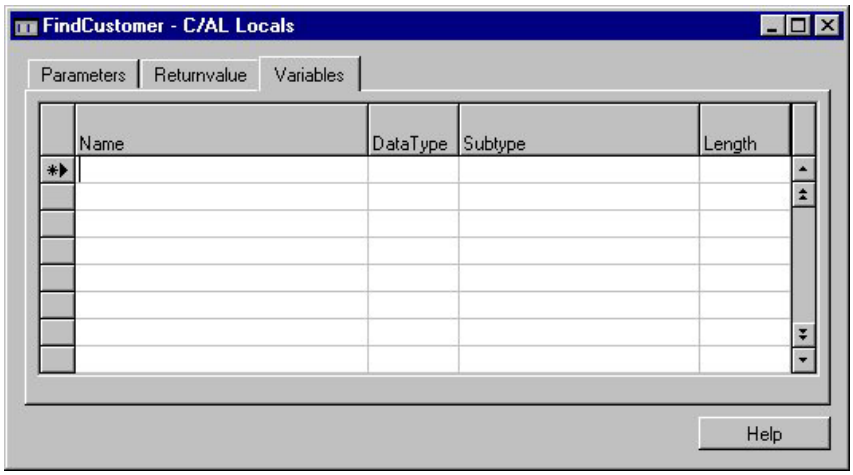
Если выбранный вами тип относится к прикладному объекту, вы также должны добавить подтип, т.е. имя конкретного объекта в БД. Если вы выбираете текст или код, необходимо определить его длину (длина кода по умолчанию составляет 10 символов, длина текста – 30 символов).

5 Щелкните закладку **Return Value**, чтобы определить возвращаемое значение для своей 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 Нажмите закладку **Variables**, чтобы определить локальные переменные. Среда C/SIDE отобразит следующее окно.



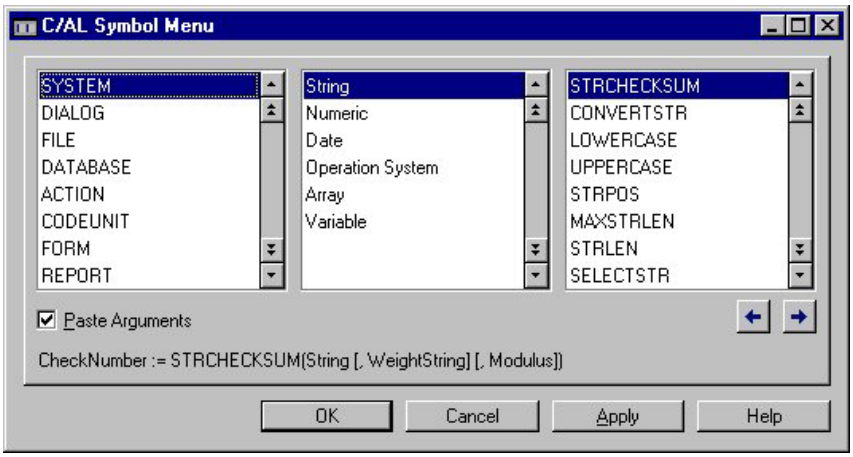
8 Вы должны добавить имя и тип для каждой локальной переменной. Если выбранный вами тип относится к прикладному объекту, вы также должны добавить подтип, т.е. имя конкретного объекта в БД. Если вы выбираете текст или код, необходимо определить длину переменной (длина кода по умолчанию составляет 10 символов, длина текста – 30 символов).

Использование меню C/AL Symbol Menu

При написании кода C/AL в редакторе вы можете использовать меню C/AL Symbol Menu, чтобы иметь возможность просмотра:

- всех переменных, определенных в программном модуле;
- всех функций C/AL.

Данные в меню C/AL Symbol Menu распределены по столбцам. Столбец слева показывает имена переменных (если вы их определили) и категории функций. Во втором столбце отображаются подкатегории, а в третьем (справа от изображения внизу) – функции в той категории, которую вы выбрали. Синтаксис отображается в нижней части окна, и вы можете нажать кнопку OK или Apply, чтобы приложение C/SIDE вставило строку в Редактор C/AL-кода.



В некоторых случаях – например, когда элемент управления на форме является дополнительной формой (subform) или когда поле является BLOB-полем – в окне отображается более трех столбцов. Вы можете использовать правую кнопку со стрелкой для прокрутки вправо столбцов меню символов (и левую кнопку со стрелкой, чтобы прокрутить их в обратном направлении).

При нажатии на кнопку OK окно **Symbol Menu** автоматически закрывается; при нажатии на кнопку Apply оно 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-зависимое «Справочное руководство по C/SIDE» (C/SIDE Reference Guide).

Компиляция и сохранение программных модулей

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.

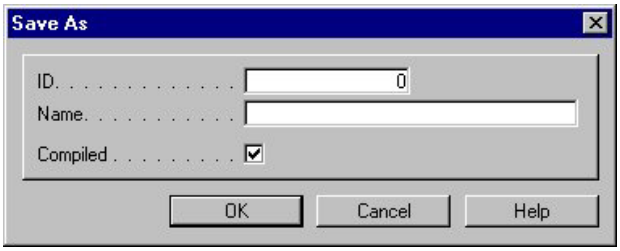
Чтобы откомпилировать код в программном модуле, выполните следующие действия.

1 Нажмите Tools, Compile.

2 Если система обнаружит ошибки в коде, она выведет на экран сообщение. Исправьте ошибку и снова нажмите Compile.

Чтобы сохранить программный модуль, выполните следующие действия.

1 Нажмите File, Save, и приложение C/SIDE отобразит следующее окно.



Введите идентификационный номер и имя. Номер используется для идентификации, а имя служит в качестве метки. 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.

6.3 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:

```
<CodeunitVariableName>.<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 этим функциям из формы.

Программный модуль с именем StatisticsFunctions

`F(x:integer)`
`Begin`
`  ...`
`End`

`G(x:integer)`
`Begin`
`  ...`
`End`

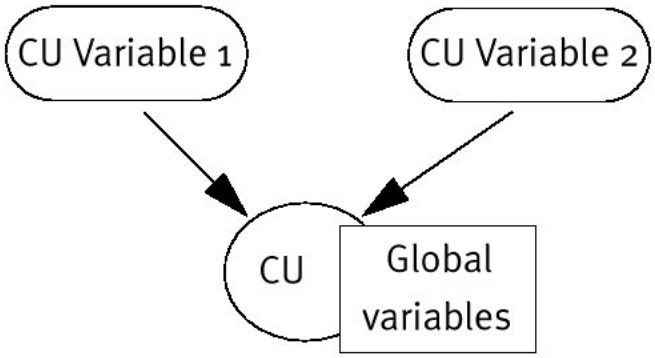
Любая форма (глобальная переменная StatFun объявлена как программный модуль с подтипом StatisticsFunctions)

`...`
 	
`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.

Пример

Программный модуль 50000 имеет две функции - Set и Get. Функция Set присваивает внутренней переменной значение данного параметра. Функция Get возвращает значение внутренней переменной.

```
VAR
  CoMIC1: Codeunit 50000;
  CoMIC2: Codeunit 50000;

//программный модуль 50000 – стандартный программный
//модуль
BEGIN
  CoMIC1.Set(1);
  CoMIC2.Set(2);
  //обеим переменным программного модуля присваиваются
  //значения - у них разные экземпляры

  CoMIC1.Get();
  CoMIC2.Get();
  //CoMIC1 возвращает 1; CoMIC2 возвращает 2

  CoMIC2 := CoMIC1;
  //CoMIC2 присваивается CoMIC1, и теперь они используют
  //один и тот же экземпляр

  CoMIC1.Get();
  CoMIC2.Get();
  //обе переменные программного модуля возвращают 1
END;
```

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 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.

Единственный экземпляр программного модуля

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 *Yes*, 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, которые не обладают набором свойств SingleInstance), удаляются в случаях, когда последняя переменная программного модуля, использующая данный экземпляр программного модуля, выходит за пределы области видимости. Тем не менее, единственные экземпляры программного модуля остаются действующими до тех пор, пока вы не закроете группу (company).

Пример

Программный модуль 50001 имеет две функции - `Set` и `Get`. Функция `Set` присваивает внутренней переменной значение данного параметра. Функция `Get` возвращает значение внутренней переменной. Программный модуль 50001 обладает свойством SingleInstance.

```
VAR
  CoSIC1: Codeunit 50001;
  CoSIC2: Codeunit 50001;

//программный модуль 50001 – единственный экземпляр
//программного модуля
BEGIN
  CoSIC1.Set(7);
  //создается экземпляр программного модуля, если он не
  //существовал раньше

  CoSIC2.Get();
  //возвращает 7 - CoSIC2 использует тот же экземпляр,
  //что и CoSIC1, поэтому они используют одни и те же
  //внутренние переменные
END;
```

Примечание

Допустимо использование единственного экземпляра программного модуля в нескольких объектах.

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).

Особые замечания по поводу Navision Application Server

Чтобы полностью использовать возможности сервера Navision Application Server, необходимо модифицировать Codeunit 1. При запуске Navision Application Server он ведет поиск триггера с идентификатором 99 в Codeunit 1. Обратите внимание, что User Portal Application Server ведет поиск триггера с идентификатором 98. При запуске сервера Navision Application Server есть параметр, который передается всегда исполняемому триггеру программного модуля. Этот параметр представлен в виде строки длиной не более 1024 символов. Интерпретация этой строки возлагается на C/AL-программиста. Если вы не предоставите параметр, процедура запуска сервера Navision Application Server завершится ошибкой.

6.4 Обзор главы

В этом разделе содержатся вопросы, касающиеся материала данной главы. Ответы на вопросы можно найти в подразделе «Ответы».

Вопросы

1 Какие триггеры имеются в каждом программном модуле?

2 Какую клавишу нужно нажать, чтобы открыть Symbol Menu?

3 Что нужно прежде всего сделать, чтобы вызвать функцию в программном модуле из другого объекта?

4 Могут ли две переменные программного модуля относиться к одному и тому же экземпляру программного модуля?

5 Могут ли две переменные программного модуля, объявленные в одном и том же объекте, относиться к разным экземплярам программного модуля?

6 Как нужно изменить переменную программного модуля, чтобы она указывала на тот же экземпляр программного модуля, что и другая переменная модуля?

7 Какую функцию необходимо использовать, чтобы переменная программного модуля указывала на новый экземпляр модуля?

8 Если программный модуль сконструирован как единственный экземпляр программного модуля, могут ли две переменные программного модуля относиться к разным экземплярам данного модуля?

9 Как можно ссылаться на глобальную переменную программного модуля из объекта, имеющего переменную с именем CU для этого модуля?

10 Можете ли вы создать функцию с именем Power в программном модуле?

Ответы

1 Какие триггеры имеются в каждом программном модуле?

Триггер Documentation используется для документирования программного модуля. Триггер OnRun вызывается, когда вы запускаете программный модуль.

2 Какую клавишу нужно нажать, чтобы открыть Symbol Menu?

F5

3 Что нужно сделать прежде всего, чтобы вызвать функцию в программном модуле из другого объекта?

Нужно создать переменную для программного модуля с правильным подтипом.

4 Могут ли две переменные программного модуля относиться к одному и тому же экземпляру программного модуля?

Да.

5 Могут ли две переменные программного модуля, объявленные в одном и том же объекте, относиться к разным экземплярам программного модуля?

Да.

6 Как нужно изменить переменную программного модуля, чтобы она указывала на тот же экземпляр программного модуля, что и другая переменная модуля?

Путем присваивания

CU1 := CU2;

7 Какую функцию необходимо использовать, чтобы переменная программного модуля указывала на новый экземпляр модуля?

CLEAR(CU1);

8 Если программный модуль сконструирован как единственный экземпляр программного модуля, могут ли две переменные программного модуля относиться к разным экземплярам данного модуля?

Нет.

9 Как можно ссылаться на глобальную переменную программного модуля из объекта, имеющего переменную с именем CU для этого модуля?

Вы не можете за пределами модуля ссылаться на его глобальные переменные. Вам придется в программный модуль добавить функции, которые будут возвращать или изменять значение переменных.

10 Можете ли вы создать функцию с именем Power в программном модуле?

Нет. Power уже является встроенной функцией.