Архитектура Navision

NAVISION®
Academy

Архитектура 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-004-v.01.00-RURU

Оглавление

Глава 1.

Введение **1-1**

1.1 Структура учебного курса 1-2

1.2 Клиент-серверная среда системы Navision 1-3

Глава 2.

Установка **2-1**

2.1 Способы установки клиентского и серверного ПО 2-2

2.2 Установка клиентского ПО для работы в автономном режиме 2-8

2.3 Установка Navision Server 2-10

2.4 Программные параметры 2-13

2.5 Контрольные вопросы 2-22

Глава 3.

Стандартная база данных системы Navision **3-1**

3.1 Содержимое базы данных 3-2

3.2 Организация информации в базе данных 3-3

3.3 Создание файла стандартной базы данных 3-5

3.4 Основные свойства стандартной СУБД Navision 3-7

Глава 4.

Опция SQL Server **4-1**

4.1 Преимущества использования СУБД SQL Server 4-2

4.2 Как работает база данных в формате SQL Server 4-3

Глава 5.

Обеспечение безопасности в системе Navision **5-1**

5.1 Обзор возможностей системы безопасности 5-2

5.2 Введение в объекты 5-4

5.3 Ограничение доступа 5-6

5.4 Определение прав доступа 5-14

5.5 Изменение пароля 5-17

5.6 Настройка учетных записей пользователей Windows 5-18

5.7 Несколько слов о лицензировании 5-19

Глава 6.

Функциональные модули и гранулы **6-1**

6.1 Функциональные модули Navision 6-2

6.2 Основные компоненты функционального модуля 6-4

6.3 Стандартные зависимости в функциональном модуле 6-13

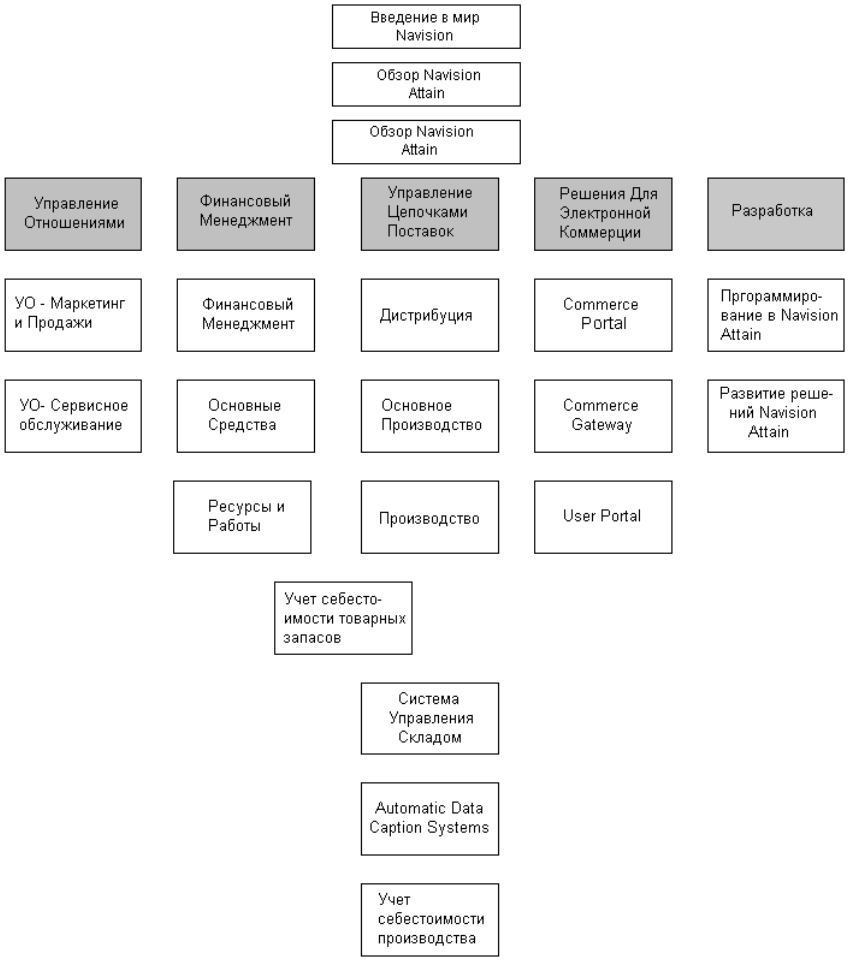
6.4 Перемещение по функциональному модулю 6-16

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

После прочтения настоящей главы вы получите представление о клиент-серверной модели, используемой в системе Navision®.

Данная глава содержит общую информацию, касающуюся архитектуры системы. Установка и настройка клиент-серверной среды описывается в последующих главах.

Настоящая глава состоит из следующих разделов:

- Структура учебного курса;
- Клиент-серверная среда системы Navision.

1.1 Структура учебного курса

Настоящий курс предназначен для тех, кто желает понять принципы, лежащие в основе архитектуры Navision. Перед установкой и настройкой системы рекомендуется изучить это пособие.

Данный курс позволит вам получить общее представление о системной и прикладной архитектуре Navision. Вы узнаете о том, как клиентская часть в Navision взаимодействует с серверной, а также о том, как устанавливать серверное и клиентское программное обеспечение. Последние несколько глав посвящены структуре приложения.

В этом курсе представлены следующие главы:

- Глава 1. Введение — настоящая глава
- Глава 2. Установка клиентского и серверного ПО
- Глава 3. Стандартная база данных
- Глава 4. Опция SQL Server
- Глава 5. Модель безопасности Navision
- Глава 6. Функциональные зоны и гранулы

В каждой главе приведено примерное время, необходимое для ознакомления с представленным в ней материалом. Помните, что этот курс предназначен для самостоятельного изучения. Постарайтесь как можно более внимательно прочитать содержащуюся в нем информацию. Выполняйте упражнения вдумчиво, не спеша. Если вам необходимо вернуться к более ранней главе, сделайте это.

1.2 Клиент-серверная среда системы Navision

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

Navision может использовать две базы данных: стандартную БД в формате Navision и базу данных в формате SQL Server корпорации Microsoft.

Вы можете использовать встроенную клиентскую СУБД клиента для управления стандартной базой данных в формате Navision при работе в автономном режиме. СУБД также поставляется и в серверном варианте, что позволяет нескольким клиентским системам одновременно подключаться к общей базе данных. Этот сервер работает на выделенном компьютере, с которым взаимодействуют все клиентские системы.

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

Взаимодействие между клиентом и сервером осуществляется одним из трех способов, перечисленных ниже. При использовании SQL Server для обмена информацией между клиентом и сервером применяется интерфейс ODBC. При использовании Navision Server связь между клиентом и сервером осуществляется по протоколам TCP/IP или NetBIOS.

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

Сервер приложений Navision Application Server (NAS) — это сервер промежуточного уровня, поддерживающий многозвенную архитектуру. Он позволяет выполнять бизнес-логику без вмешательства пользователя. При помощи Navision Application Server можно организовать взаимодействие с внешними сервисами.

Navision Application Server выступает в роли клиента по отношению к

серверу баз данных и может служить сервером для других сервисов. При запуске Navision Application Server открывает заданную базу данных и исполняет код C/AL™ в заданном программном модуле.

Navision Application Server может взаимодействовать с Navision Server и с SQL Server тем же способом, что и клиентская часть Navision. Navision Application Server поддерживает только механизм Windows-аутентификации и автоматически восстанавливает связь с сервером баз данных в случае проблем с сетью.

Клиент

Раннее говорилось о том, что основное предназначение клиента Navision — предоставить пользовательский интерфейс, однако его возможности гораздо шире. В автономном режиме, упомянутом выше, клиент способен напрямую подключаться к файлу стандартной базы данных, минуя сервер. Кроме того, клиент отвечает за исполнение всей бизнес-логики. Он считывает из базы данных объекты и обеспечивает контроль за их функционированием. Основная часть приложения Navision работает на индивидуальных клиентских системах.

Сервер

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

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

Это неполный перечень. Он приведен лишь для того, чтобы дать общее представление о функциях сервера. Microsoft SQL Server также

выполняет все эти задачи. Однако стандартный сервер, в отличие от SQL Server, осуществляет слежение за различными версиями одной записи. Речь об этом пойдет в последующих главах.

Связка “клиент -- сервер баз данных” представляет собой полностью интегрированное решение.

Глава 2

Установка

Прочитав эту главу, вы узнаете, как устанавливать клиентское и серверное ПО системы Navision.

Эта глава состоит из следующих разделов:

- Способы установки клиентского и серверного ПО;
- Установка клиентского ПО для работы в автономном режиме;
- Установка Navision Server;
- Программные параметры.

2.1 Способы установки клиентского и серверного ПО

Существует несколько способов установки клиентского и серверного ПО. В этой главе мы рассмотрим различные конфигурации клиента и сервера. Будут также приведены примеры наиболее часто встречающихся ошибок.

Для успешной установки необходимо:

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

Меры предосторожности для любого типа установки

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

- не используйте утилиты сжатия данных, такие как DoubleSpace, Stacker или DriveSpace;
- ограничивайте количество сервисов и программ, работающих одновременно, они замедляют обработку данных;
- не используйте Lazy Write, Swap Disk для OS/2, Smartdrive и подобные им программы кэширования.

Примечание

В данном курсе предполагается, что в качестве операционной системы используется Windows 2000. Однако сервер также может работать под управлением Windows NT, а в качестве операционной системы для клиентских компьютеров могут использоваться Windows XP, Windows NT и Windows 98.

Установка клиентской части для работы в автономном режиме

Наиболее простой является однопользовательская конфигурация, когда клиент работает в автономном режиме. Процедура установки в этом случае предельно проста. Необходимо установить клиентское

программное обеспечение на компьютере пользователя. Затем вы должны заменить файл database.fdb базой данных, которая была создана специально для этого пользователя. (Процедура создания базы данных описана в другом курсе.) И, наконец, замените файл с лицензией fin.flf, который был установлен вместе с исполняемыми файлами, файлом с клиентской лицензией.

Установка клиентского и серверного ПО в стандартной конфигурации

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

После установки клиентского ПО на сервере можно запускать программу установки серверного ПО. Это отдельная программа. Дополнительная информация о ней содержится в разделе 2.3 "Установка Navision Server". Программа установки попросит указать место расположения базы данных и файла с лицензией. Выберите базу данных и файл с клиентской лицензией. Программа также создаст сервис и запустит его в конце процедуры установки. Именно этот сервис отвечает за функционирование стандартного сервера Navision. После завершения процедуры установки этот сервер будет находиться в рабочем состоянии.

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

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

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

В этом примере `servername` задает имя компьютера, на котором установлен Navision Server, а `nettype` задает используемый протокол. При использовании этих параметров для создания ярлыка настройки будут сохранены, и пользователю потребуется помнить лишь собственный пароль.

Теперь щелкните ярлык дважды, чтобы протестировать соединение. Ошибки чаще всего возникают из-за того, что клиент не может установить соединение с сервером. В зависимости от используемого протокола существует несколько способов идентификации проблемы. (См. раздел «Устранение проблем при установке соединения с базой данных».)

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

Установка опции Microsoft SQL Server для использования с Navision

Процедура установки при использовании опции SQL Server в корне отличается от описанной выше. Установите ПО SQL Server на серверном компьютере в соответствии с рекомендациями корпорации Microsoft.

В этом случае необходимо установить на серверном компьютере лишь один компонент с компакт-диска Navision — расширенные хранимые процедуры (extended stored procedures). Запустите установочный компакт-диск Navision и в папке $:\SQL_ESP (где $ обозначает букву, соответствующую приводу CD-ROM) щелкните файл `xp_ndo.exe`. Этот файл содержит расширенные хранимые процедуры. Укажите программе путь к вложенной папке `BINN`, находящейся в той папке, где установлено ПО SQL Server. Распакованный файл `xp_ndo.dll` должен быть помещен в эту папку на сервере.

Теперь осталось только установить клиентское ПО. После того, как вы установили SQL Server и убедились, что он нормально функционирует, можно приступать к установке Navision на клиентских компьютерах. Эта процедура была описана в разделе «Установка

клиентского и серверного ПО в стандартной конфигурации». Единственное отличие в том, что вместо исполняемого файла `fin.exe` нужно использовать `finsql.exe`.

Когда клиент Navision устанавливает соединение с сервером в первый раз, программа просит установить лицензию на сервере. Более того, она автоматически добавит расширенные хранимые процедуры. Если же программа обнаружит, что доступ к серверу с помощью Navision уже осуществлялся ранее, то расширенные хранимые процедуры автоматически добавлены не будут, и вам придется сделать это вручную.

Чтобы добавить расширенные хранимые процедуры вручную, требуется сделать следующее.

Используйте средства корпорации Microsoft, например, Enterprise Manager, чтобы добавить файл `xp_ndo.exe` к расширенным хранимым процедурам, уже установленным в составе ПО SQL Server. Название расширенной хранимой процедуры *xp_ndo_enumusersids*.

Установка нескольких стандартных серверов

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

При использовании протокола NetBIOS нужно просто использовать различные имена сервера, что позволит обоим сервисам взаимодействовать с любым клиентом. На клиентской стороне необходимо изменить значение параметра *servername* для ярлыка, чтобы клиент мог идентифицировать сервис, с которым необходимо установить соединение.

При использовании протокола TCP/IP дело обстоит немного сложнее. В этом случае необходимо изменить номера портов для сервисов. Чтобы изменить номер порта, нужно отредактировать файл `services` на серверном компьютере. Обычно этот файл находится в каталоге `C:\WINNT\system32\drivers\`.

Если вы не знаете, как редактировать этот файл, обратитесь к специалисту, знакомому с TCP/IP. После того, как необходимые

изменения были произведены на сервере, следует сделать это для каждого клиента. Также вы можете изменить ярлык на каждом клиентском компьютере, чтобы задать имя сервера.

Примечание

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

Устранение проблем при установке соединения с базой данных

Проблема может возникнуть, если стандартный сервер Navision функционирует нормально, но клиентские компьютеры не могут установить соединение с ним. Способ устранения этой проблемы зависит от используемого протокола.

NetBIOS

Чтобы определить, почему клиенты Navision не могут установить связь со стандартным сервером по протоколу NetBIOS:

1) убедитесь в том, что протокол NetBIOS используется по умолчанию на серверном компьютере;
2) убедитесь в том, что протокол NetBIOS используется по умолчанию на клиентском компьютере.

Если проблему устранить не удалось, обратитесь в службу поддержки.

TCP/IP

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

1 Запустите клиентское ПО Navision и попытайтесь установить соединение со стандартным сервером, используя имя сервиса (без префикса «Navision Server»).

Если это удалось, клиент может установить соединение с сервером и сервисом по протоколу TCP/IP, используя имя сервиса. Если это не удалось, Navision не может использовать TCP/IP. Возможно, проблема связана с файлом `hosts`.

Переходите к следующему пункту.

2 Используйте утилиту ping для проверки доступности сервера по IP-адресу (в командной строке введите: ping ###.###.###.###).

Если это удалось, данный клиентский компьютер может установить соединение с серверным компьютером. Переходите к пункту 3. Если это не удалось, проблема связана с установкой соединения с серверным компьютером по TCP/IP, а не с ПО Navision. Перед тем, как продолжить, необходимо устранить эту проблему.

3 Используйте утилиту ping для проверки доступности сервиса по имени (без префикса “Navision Server”; в командной строке введите: ping servicename). Учтите, что на самом деле вы не проверяете доступ к сервису. Вы по сути проверяете другим способом доступ к серверу.

Если это удалось, файл `hosts` сконфигурирован корректно.

Если это не удалось, значит в файле `hosts` отсутствует запись, позволяющая стеку TCP/IP определить, где работает сервис. Добавьте в файл `hosts` строку, содержащую имя сервиса (без префикса «Navision Server») и IP-адрес сервера. Теперь остается лишь протестировать соединение между сервером и клиентами.

2.2 Установка клиентского ПО для работы в автономном режиме

Следует как можно более точно следовать инструкциям, приведенным в данном разделе. Это относится и к структуре каталогов для установки ПО. Клиентское ПО Navision, которое вы сейчас установите, будет использоваться на протяжении всего учебного курса. Компакт-диск Navision содержит программы установки как клиентского ПО, так и серверного ПО Navision Server. При установке клиентского ПО следует установить также демонстрационную базу данных, содержащую раздел для вымышленной компании.

Установка клиентского ПО

Windows XP, 98, 2000, NT 4.0

Вставьте установочный компакт-диск. В браузере автоматически откроется HTML-страница, содержащая следующие опции:

- **Demo (Демо)** – установка демонстрационной версии клиента Navision;
- **Client (Клиент)** – установка клиента Navision;
- **Servers (Серверы)** – установка сервера Navision;
- **Tools (Средства)** – установка средств Navision для C/ODBC, C/OCX и C/Front;
- **Overview (Обзор)** – обзор содержимого компакт-диска;
- **Home (Главная страница)** – возврат к главному меню.

Щелкните опцию Client, затем выберите Installation of Navision Client for Windows 98, Windows NT Windows 2000. Запустится стандартная программа установки программного обеспечения ОС Windows. Далее следуйте ее указаниям.

Файлы на жестком диске

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

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

2.3 Установка Navision Server

Перед установкой Navision Server для пользователя, следует как можно больше практиковаться.

В этом разделе описывается установка сервера на компьютере под управлением ОС Windows 2000. Это наиболее распространенная конфигурация.

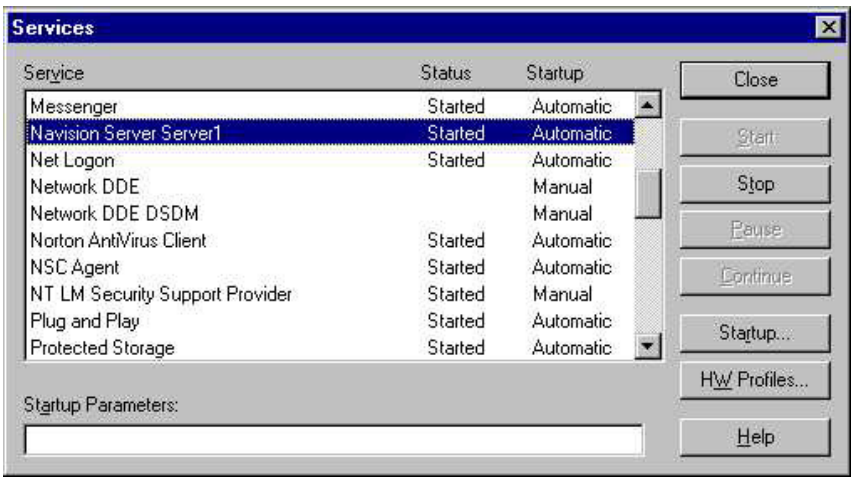
Вставьте установочный компакт-диск. В браузере автоматически откроется HTML-страница, содержащая следующие опции:

- **Demo (Демо)** – установка демонстрационной версии клиента Navision;
- **Client (Клиент)** – установка клиента Navision;
- **Servers (Серверы)** – установка сервера Navision;
- **Tools (Средства)** – установка средств Navision для C/ODBC, C/OCX и C/Front;
- **Overview (Обзор)** – обзор содержимого компакт-диска;
- **Home (Главная страница)** – возврат к главному меню.

Щелкните опцию Servers, затем выберите Navision Server for Windows NT 4.0 (Intel) and Windows 2000. Запустится стандартная программа установки программного обеспечения ОС Windows. Далее следуйте ее указаниям.

Navision Server может быть запущен как сервис или как приложение.

После установки сервиса на компьютере под управлением ОС Windows 2000, вы можете увидеть его в окне **Services**:



Преимущества такого способа запуска сервера будут описаны позднее.

На рисунке, приведенном выше, Navision Server — это сервис, который был установлен программой установки. Он автоматически запускается в процессе установки и настроен таким образом, чтобы стартовать каждый раз при включении компьютера.

В качестве приложения

Ниже приводится содержимое окна командного процессора, где Navision Server был запущен в качестве приложения. Сначала убедитесь в том, что сервер не запущен в качестве сервиса. Перейдите в каталог, где расположены файлы серверного ПО и введите «Server». После этого клиенты могут устанавливать соединение с сервером, как если бы он был запущен в качестве сервиса.

```
D:\Program Files\Navision Server 2.60>server
Database server for Navision version 3.01 (Build 7700)
Copyright (c) 1987-2001 Navision a/s
Starting database server SERVER1 ...
Cache(Kb)...........: 70144
CommitCache.........: No
Maximum sessions....: 2
Maximum database(Mb): 64000
NetType.............: tcp
Database............:C:\\NavisionAttain\\Database Server\\database.fdb
Licensed to.........: CRONUS International Ltd.
                      John Roberts
                      The Ring
                      Westminster
                      W2 8HG London
                      World Wide 1
                      0666-666-6666
```

Запуск из командной строки имеет свои преимущества. Как видите, все параметры сервера выводятся на экран, чего не происходит при запуске в качестве сервиса. Кроме того, при использовании командной строки сообщения об ошибках выдаются в развернутом виде.

Убедившись таким образом, что все работает, вы можете быть уверены в том, что сервис будет функционировать без проблем.

Преимущества функционирования Navision Server в качестве сервиса

Вы можете запустить Navision Server двумя способами: в качестве сервиса Windows или в качестве приложения. При функционировании сервера в качестве приложения легче следить за его состоянием и контролировать его параметры. Однако использование его в качестве сервиса Windows дает гораздо больше преимуществ.

Если Navision Server используется в качестве сервиса Windows, по умолчанию он настроен таким образом, чтобы стартовать при загрузке ОС Windows 2000. Поэтому Navision Server автоматически перезапускается при перезапуске Windows 2000.

Если сервер используется в качестве приложения, кто-то должен открыть интерактивную сессию на сервере под управлением Windows 2000 и запустить программу. Этот пользователь не может завершить сессию, поскольку при этом все программы этой сессии также завершаются. (К сервисам Windows 2000 это не относится.) Так как пользователь не может завершить сессию на консоли сервера, это облегчает несанкционированный доступ к серверу и данным, хранящимся на нем.

Если же вы запускаете Navision Server в качестве сервиса, программа работает в фоновом режиме и не зависит от сессий пользователей на консоли сервера. Это обеспечивает более высокий уровень безопасности сервера под управлением ОС Windows 2000.

2.4 Программные параметры

Система Navision позволяет задавать значения ряда программных параметров. Некоторые из этих параметров относятся к серверу, остальные — к клиенту.

Программные параметры сервера

CACHE

- Устанавливает объем областей кэширования DBMS Cache и Commit Cache, используемых Navision.

Максимальный объем 1 Гбайт.

Рекомендуется устанавливать объем DBMS Cache таким образом, чтобы на каждого пользователя приходилось 2 Мбайт. При этом общий объем не должен превышать 900 Мбайт, и по меньшей мере 128 Мбайт оперативной памяти должно оставаться в распоряжении операционной системы.

Это повышает производительность, поскольку для работы с базой данных отводится определенное количество памяти.

Чтение данных из оперативной памяти происходит быстрее, чем чтение с жесткого диска.

Область кэширования делится на DBMS Cache (чтение) и Commit Cache (запись). Navision автоматически распределяет память между этими областями с целью оптимизации.

COMMITCACHE

- Активирует Commit Cache.

Это ускоряет вставку, модификацию и удаление данных, поскольку сервер производит запись в оперативную память вместо того, чтобы обращаться непосредственно к жесткому диску.

Объем памяти, отводимый для Commit Cache, составляет от 1/3 до 2/3 полного объема области кэширования. Когда объем Commit Cache достигает 2/3 полного объема, Navision принудительно устанавливает для транзакций записи более высокий приоритет, чем для транзакций чтения.

Кроме того, в отсутствие транзакций чтения, ожидающих обработки, завершенные транзакции записи будут записаны на жесткий диск.

DATABASE

- Задает при запуске сервера путь к файлу базы данных и его имя.

INSTALLASSERVICE

- Добавляет сервер в список сервисов.

После применения этой команды необходимо использовать окно **Services** для запуска и остановки Navision Server.

UNINSTALLASSERVICE

- Удаляет сервер из списка сервисов.

После применения этой команды нельзя будет использовать окно **Services** для запуска и остановки сервера. Сервер можно будет запускать только как приложение.

STOPTIME

- Останавливает сервер автоматически в определенное время.

Сервер автоматически остановится, когда встроенные часы компьютера покажут заданное время. Вы можете использовать эту функцию, чтобы по истечении определенного времени прекратить доступ к базе данных.

NETTYPE

- Задает сетевой протокол, который будет использоваться для обмена данными между сервером и клиентами.

Существует два значения:

- TCP для TCP/IP;
- NETB для NetBIOS.

Значение этого параметра должно быть одинаковым как для всех клиентов, так и для сервера.

SERVERNAME

- Задает имя сервера, которое должны использовать клиенты для установки соединения.

SESSIONS

- Устанавливает максимальное количество соединений, поддерживаемых сервером одновременно.

Navision позволяет поддерживать от 1 до 500 сессий. Однако максимальное количество сессий может быть ограничено используемым протоколом или лицензией.

Примеры

Следующая команда запускает сервер как приложение, задает имя сервера *MyServer* и сетевой протокол TCP. Сервер автоматически остановится в полночь.

```
SERVER SERVERNAME=MyServer,NETTYPE=TCP, STOPTIME=240000
```

Следующая команда добавляет сервер в список сервисов, задает имя сервера *MyServer* и сетевой протокол TCP.

```
SERVER SERVERNAME=MyServer,NETTYPE=TCP,INSTALLASSERVICE
```

Следующая команда удаляет сервер с именем *MyServer* из списка сервисов.

```
SERVER SERVERNAME=MyServer,UNINSTALLASSERVICE
```

Следующая команда изменяет объем области кэширования и добавляет сервер с именем *MyServer* в список сервисов.

```
SERVER SERVERNAME=MyServer,CACHE=50000,INSTALLASSERVICE
```

Программные параметры клиента

CACHE

- Предназначение этой опции то же, что и для сервера.

Если клиент соединяется с базой данных на сервере, этот параметр игнорируется.

COMMITCACHE

- Предназначение этой опции то же, что и для сервера.

Если клиент соединяется с базой данных на сервере, этот параметр игнорируется.

COMPANY

- Задает название компании, чей раздел базы данных открывается автоматически.

DATABASE

- Задает при запуске Navision путь к файлу базы данных и его имя.

Если клиент устанавливает соединение с сервером, эта опция игнорируется, так как путь к базе данных устанавливается сервером.

DBREADONLY

- Эта опция позволяет открыть базу данных в режиме «только чтение». В этом режиме запрещено изменение, добавление или удаление записей.

Применяется только в случае однопользовательского доступа к базе данных.

Запустить сервер в режиме «только чтение» невозможно.

Этот параметр может использоваться для устранения неисправностей в случае, если не удается открыть базу данных в нормальном режиме.

ID

- Позволяет пользователю зарегистрироваться в системе, используя другой .ZUP-файл.

Значение этого параметра по умолчанию FIN.

NETTYPE

- Задает сетевой протокол, который будет использоваться для соединения с сервером.

Существует два значения:

- TCP для TCP/IP;
- NETB для NetBIOS.

Сервер должен использовать тот же протокол.

OBJECTCACHE

- Устанавливает объем области кэширования, которую клиентский компьютер использует для хранения описаний объектов.

Объем области кэширования зависит от общего объема оперативной памяти и количества задач, регулярно выполняемых клиентским компьютером.

Для этой цели рекомендуется выделить от 1 Мбайт до 12 Мбайт оперативной памяти. При этом по меньшей мере 64 Мбайт должно остаться в распоряжении операционной системы. Если общий объем оперативной памяти невелик, объем ObjectCache может быт уменьшен до 500 Кбайт, но в этом случае производительность снизится.

SERVERNAME

- Задает имя сервера, с которым клиент должен установить связь при запуске.

NTAuthentication

- Задает тип механизма аутентификации, используемый при регистрации на сервере и открытии базы данных.

Возможное значение:

- yes

Если вы используете механизм Windows-аутентификации, вы можете открыть заданную базу данных, не вводя имя пользователя и пароль.

DBTEST

- Производит проверку целостности базы данных

Возможные значения:

- min

- normal

- max

Эти значения соответствуют различным типам тестов для баз данных, которые могут быть проведены с помощью Navision.

TEMPPATH

- Задает путь к папке для хранения временных файлов.

По умолчанию этот параметр содержит путь к папке, где расположен файл `fin.exe`.

Если файл `fin.exe` находится на удаленном диске, задайте с помощью TEMPPATH путь к папке на локальном жестком диске. Это позволит увеличить производительность.

Если папка, в которой находится `fin.exe` , используется совместно несколькими пользователями, то для предотвращения конфликтов задайте с помощью TEMPPATH путь к папке на локальном жестком диске для каждого пользователя.

Примеры

Эта команда запускает клиента и устанавливает соединение с сервером *MyServer* по протоколу TCP:

```
FIN SERVERNAME=MyServer,NETTYPE=TCP
```

Эта команда запускает клиента, задает другое имя ZUP-файла и помещает его на удаленный диск Z:

```
FIN ID=Z:\UserPreferences\Sally.ZUP
```

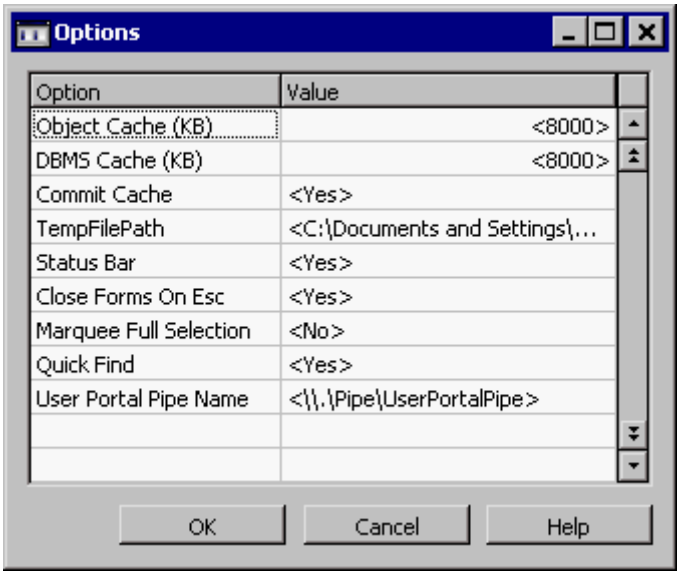
Эта команда запускает клиента и открывает базу данных на локальном жестком диске (в автономном режиме). База данных открывается в режиме «только чтение».

```
FIN DBREADONLY=Yes,DATABASE=C:\Databases\Database1.fdb
```

Окно Options

Значение большинства параметров, описанных выше, может быть установлено с помощью окна клиентского приложения. Например, вы можете открыть базу данных, выбрав в меню File пункт Database, и затем Open.

Многие другие опции для настройки клиента можно найти, открыв окно **Options**. Для этого выберите в меню Tools пункт Options.



В таблице, приведенной ниже, дано описание каждой опции и ее значение по умолчанию.

Опция	Значение по умолчанию
Object Cache (Кбайт)	8000
Опция Object Cache включает кэширование объектов программы Navision.	
DBMS Cache (Кбайт)	8000
Опция DBMS Cache задает объем памяти, используемый для кэширования данных (максимум 1000 Мбайт). Если эта опция задается для клиента в меню Client Options, она имеет силу только в автономном режиме. Для того чтобы изменения вступили в силу, необходимо перезапустить программу.	
Commit Cache	Yes

Опция	Значение по умолчанию
	Эта опция включает кэширование операций записи. Возможные значения Yes/No. Если эта опция задается для клиента в меню Client Options, она имеет силу только в автономном режиме. Для того чтобы изменения вступили в силу, необходимо перезапустить программу.
TempFilePath	C:\TEMP\
	TempFilePath — это каталог, используемый Navision для записи временных файлов. При задании нового пути с помощью опции TempFilePath, расположение каталога, показанного здесь, изменится.
Status Bar	Yes
	Эта опция активирует или деактивирует панель состояния (Status Bar)
Close Form On Esc	Yes
	Эта опция определяет, будут ли закрываться окна при нажатии клавиши ESC.
Marquee Full Selection	No
	Эта опция определяет способ выделения графических объектов на экране. Эта настройка особенно полезна для пользователей, работающих в среде разработки C/SIDE. Если выбрано значение **Yes** , для выделения группы объектов необходимо нарисовать прямоугольник, содержащий их целиком. Если выбрано значение **No**, для выделения объекта достаточно нарисовать прямоугольник, касающийся его.
Quick Find	Yes
	Эта опция определяет, будет ли автоматически вызываться окно поиска (Find), когда вы набираете текст или поле исходного текста, которое нельзя редактировать. Если выбрано значение *Yes* , окно поиска (Find) будет вызываться автоматически. Если выбрано значение *No* , вы можете открыть окно поиска (Find) только вручную.

Резюме

Теперь у вас есть достаточно хорошее представление о процессе установки Navision Server и клиентского ПО. Приведенные здесь примеры весьма просты. В зависимости от потребностей вашего заказчика, конфигурация может быть гораздо более сложной, однако ваших знаний достаточно для того, чтобы приступить к работе.

2.5 Контрольные вопросы

Упражнение 1 **Планирование установки**

У вашего заказчика 100 сотрудников, которые будут использовать Navision. Они работают в три смены. Каждый сотрудник использует компьютер совместно с двумя другими, работающими в разные смены. С помощью каждого компьютера устанавливается только одно соединение. В качестве сетевого протокола заказчик использует TCP/IP. Составьте план, в соответствии с которым вы будете устанавливать Navision для этого заказчика. Также составьте перечень оборудования, которое будет необходимо приобрести заказчику и опишите настройки сервера.

Упражнение 2 **Проблемы, связанные с доступом по сети**

Ваш заказчик использует TCP/IP в качестве сетевого протокола. После замены компьютера, на котором установлен Navision Server, на более современный клиенты не могут установить соединение с сервером. Каков будет ваш первый шаг для решения этой проблемы?

Упражнение 3 **Две версии на одном компьютере**

Вы убедили заказчика установить последнюю версию Navision вместо предыдущей. Прежде чем перейти к использованию последней версии, заказчик хочет протестировать ее. Он просит вас установить последнюю версию Navision на том же сервере, где установлена рабочая версия. Объясните, как это можно осуществить.

Глава 3

Стандартная база данных системы Navision

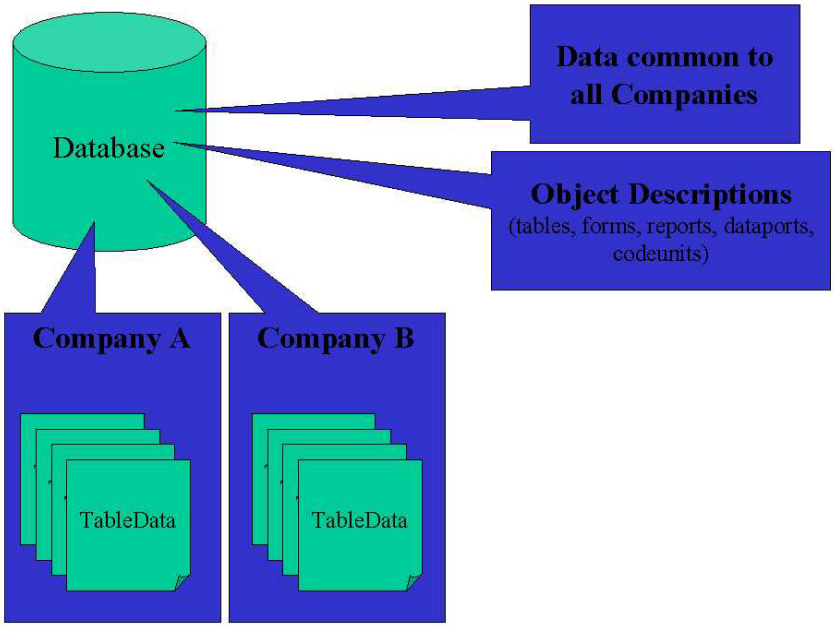
Эта глава посвящена устройству и функционированию стандартной СУБД системы Navision. Рассматривая ее, мы не будем углубляться в детали реализации СУБД. Основное внимание будет уделено понятию базы данных и возможностям, реализованным в стандартной СУБД системы Navision.

Эта глава состоит из следующих разделов:

- Содержимое базы данных;
- Организация информации в базе данных;
- Создание файла стандартной базы данных;
- Основные свойства стандартной СУБД Navision.

3.1 Содержимое базы данных

Приведенная диаграмма иллюстрирует типы информации, хранимые в базе данных:



Как видно из рисунка, основная масса информации – это обычные табличные данные. Они разделены на несколько групп. Табличные данные состоят из двух частей – части, расположенной в разделе данных конкретной компании, и другой части, расположенной в общем для всех компаний разделе базы данных.

Другой важной частью базы данных являются объекты приложений. Существует пять типов объектов: таблицы, формы, отчеты, порты данных и программные модули. Все используемые клиентскими приложениями системы Navision объекты хранятся в базе данных.

Кроме того, в базе данных хранятся ключи, не показанные на этом рисунке. Они служат для сортировки и поиска информации в БД. Хотя это не видно из рисунка, ключи составляют существенную часть общего объема базы данных.

3.2 ОРГАНИЗАЦИЯ ИНФОРМАЦИИ В БАЗЕ ДАННЫХ

Для изучения дальнейшего материала необходимы базовые знания о том, как информация хранится в базе данных. Это облегчит понимание, почему одни действия с базой данных выполняются просто, а другие – невозможны.

Объекты

Рассмотрим хранение объектов в базе данных. Каждый занесенный в БД объект хранится в таблице **Object** (Таблица с идентификатором 2000000001). Эта таблица не отображается в Дизайнере объектов, так как пользователи не могут изменять ее. Любые изменения в этой таблице могут нарушить функционирование СУБД.

При создании нового объекта или изменении существующего содержимое этой таблицы обновляется автоматически. Способ обновления этой таблицы требует, чтобы при создании и модификации объектов доступ к базе данных имел лишь один пользователь. При необходимости добавления или изменения объектов рекомендуется использовать копию базы данных, либо делать это тогда, когда остальные пользователи не имеют доступа к БД.

Сами объекты хранятся в двоичном формате в BLOB-поле (поле для хранения больших двоичных объектов – Прим. ред.) соответствующей записи БД. В таблице **Object** имеются поля **Object type** и **ID**. Именно эту таблицу отображает Дизайнер объектов.

Данные отдельной компании

Перед рассмотрением данных, общих для всех компаний, необходимо подробнее рассмотреть части базы данных, специфичные для отдельных компаний. Большая часть информации в базе данных хранится отдельно для каждой компании. Например, как показано на приведенной выше диаграмме, в БД может храниться таблица данных о клиентах компании А. Компания В располагает собственной таблицей с аналогичной информацией. Данные этих таблиц независимы: компания А не имеет доступа к данным компании В, и наоборот.

Таблицу **Customer** можно сделать общей для нескольких компаний, но это весьма трудоемкая задача. Так как данные о клиентах хранятся раздельно для разных компаний, потребовалось бы вручную

экспортировать данные каждой компании, удалять их из базы данных, изменять определение таблицы и затем импортировать данные всех компаний в общую таблицу.

Определение таблицы хранится в таблице **Object** и, вследствие этого, одинаково для всех компаний. Однако сами данные этих таблиц различны.

Общие данные для всех компаний

Некоторые таблицы в базе данных содержат информацию, общую для всех компаний. Примером такого рода информации могут служить таблицы прав доступа. Все компании имеют одинаковые права доступа к базе данных. Кроме того, вы можете создавать дополнительные таблицы, общие для всех компаний. Такие таблицы будут размещены в разделе БД, который содержит информацию, единую для всех компаний.

3.3 Создание файла стандартной базы данных

Стандартная база данных хранится в едином логическом файле. По умолчанию используется имя файла `database.fdb`. Клиентская или серверная программы соединяются с этим файлом. Логический файл может состоять не более чем из 16 физических файлов.

Зачем использовать более одного файла для хранения БД?

Производительность СУБД может быть существенно повышена, если разделить базу данных на несколько файлов и расположить эти файлы на разных физических дисках. При такой организации БД сервер будет запрашивать данные одновременно с различных дисков и получит их быстрее, чем если бы все данные были расположены на одном физическом диске. Основное время при доступе к базе данных расходуется именно на считывание данных с диска. Распределение данных по различным физическим дискам позволяет сократить время доступа.

Как создавать файлы базы данных?

Сначала необходимо создать первичный файл базы данных (первый из набора файлов, в которых будет храниться БД). Для этого нужно щелкнуть меню **File(Файл)**, **Database(База данных)**, **New(Создать)**. Рекомендуется создавать все файлы базы данных примерно одинакового размера. Размер каждого файла может варьироваться от 1 Кбайт до 64 Гбайт. Суммарный объем всех файлов, образующих базу данных, не может превышать 64 Гбайт. При этом база данных может состоять лишь из одного файла. Выбрав размер файла БД, вы можете завершить создание первичного файла БД. Введите выбранный размер файла в окне **New Database (Новая БД)** и щелкните OK. Программа создаст первичный файл базы данных.

На следующем шаге вам следует добавить остальные файлы базы данных. Откройте первичный файл БД, щелкните **File(Файл)**, **Database(База данных)**, **Expand(Расширить)**. В окне **Expand Database (Расширить БД)** вы можете выбрать расширить первичный файл базы данных. Также вы можете выбрать меню расширенных настроек **Advanced(Дополнительно)**. В окне **Advanced(Дополнительно)** можно задать пути и имена всех создаваемых файлов. Все они должны быть примерно одинакового размера. Следует помнить, что доступ к базе данных существенно ускоряется, если она организована в виде нескольких файлов, расположенных на разных физических дисках.

Серверная часть ПО должна знать лишь путь к первичному файлу базы данных. Он, в свою очередь, содержит ссылки на остальные файлы БД. Распределение информации по нескольким файлам абсолютно прозрачно для пользователя, и с его точки зрения база данных представляет собой один логический файл. После того, как вы создали файлы, образующие БД, можно восстановить ее содержимое, используя резервную копию. Сервер самостоятельно распределит данные таким образом, чтобы максимально эффективно использовать новую структуру базы данных.

После создания БД, состоящей из нескольких файлов, и перед тем, как вводить в нее новые данные или восстанавливать содержимое из резервной копии, следует завершить работу программы Navision и снова запустить ее. При вводе информации в БД система должна выдать окно с сообщением "Восстановление списка свободных блоков" и индикатором выполнения. Это означает, что вводимая в базу данных информация равномерно распределяется между файлами, образующими БД, обеспечивая, тем самым, высокую эффективность при последующем поиске и извлечении информации.

3.4 Основные свойства стандартной СУБД Navision

Поскольку система Navision рассматривает БД как единый файл заранее установленного объема, система может контролировать все выделенное ей пространство на диске. Это позволило реализовать в системе Navision ряд функций, отсутствующих в аналогичных продуктах конкурирующих компаний. К таким функциям относятся:

- управление версиями;
- запись информации в БД с возможностью отмены изменений;
- оптимистичный параллелизм;
- встроенные функции создания резервной копии и восстановления БД из резервной копии;
- контроль целостности БД;
- кэширование СУБД;
- технология суммирования индексных полей (SumIndexField);
- создание резервных копий на сервере.

Управление версиями – версионная БД

Управление версиями в стандартной СУБД Navision Server реализуется автоматически. Однако понимание, что такое версии и как они используются в стандартной СУБД Navision, необходимо для понимания функций языка C/AL при программировании клиентских приложений.

Сначала мы введем два термина – процесс и транзакция.

Концепция Navision не предусматривает использования термина “процесс”. В частности, поэтому процессы не упоминаются в руководстве *Application Designer’s Guide*. Однако введение этого термина облегчает понимание многих аспектов функционирования системы. Термин “транзакция” также полезен, так как транзакции являются существенной частью многих процессов.

Понятие процесса

В данном контексте под процессом мы будем подразумевать все действия, производимые системой Navision с момента, когда пользователь передает управление системе, и до момента, когда он снова получает управление. Например, при нажатии клавиши на клавиатуре происходит следующий короткий процесс.

1 Система Navision определяет, какая клавиша была нажата.

2 Проводятся некоторые служебные преобразования.

3 Выбранная пользователем буква выводится на экран.

4 Управление возвращается пользователю.

Примером более длинного процесса может служить запись информации в базу данных. Пользователь щелкает кнопку записи (posting button) в экранной форме журнала, и начинается процесс записи. Процесс работает в течение некоторого времени, и лишь после его окончания управление возвращается в экранную форму журнала и пользователю. Длительность процесса не играет роли. Процесс может быть очень коротким, как обработка нажатия клавиши, или длинным, как запись в базу данных.

Понятие транзакции

Транзакцией будем называть период времени в ходе процесса, в течение которого блокирована одна или более таблиц в БД. Когда таблица блокирована, другие пользователи не могут записать или изменить ее. Поэтому при создании программного кода в Navision важно стремиться максимально сократить время выполнения транзакций.

Началом транзакции считается момент блокирования первой из тех таблиц, что блокируются в ходе выполняемого процесса. Блокирование таблицы можно вызвать в явном виде, используя функцию LOCKTABLE (БЛОКИРОВАТЬ ТАБЛИЦУ), или косвенно, используя любую из перечисленных ниже функций обновления базы данных. Все эти функции блокируют таблицу перед выполнением соответствующих операций:

- RENAME (ПЕРЕИМЕНОВАТЬ)
- INSERT (ВСТАВИТЬ)
- MODIFY (ИЗМЕНИТЬ)
- MODIFYALL (ИЗМЕНИТЬ ВСЕ)
- DELETE (УДАЛИТЬ)
- DELETEALL (УДАЛИТЬ ВСЕ)

При блокировании таблицы начинается транзакция. Блокирование дополнительно одной или нескольких таблиц в то время, когда первая таблица еще блокирована, не начинает новую транзакцию. Началом транзакции считается момент блокирования первой из таблиц, блокируемых в ходе выполняемого процесса.

Транзакция заканчивается, когда все изменения вносятся в базу данных (т.е. выполняется функция COMMIT) и не могут быть отменены. Внесение обновленных в ходе транзакции данных в БД можно инициировать, непосредственно вызвав функцию COMMIT, либо оно будет выполнено автоматически при завершении процесса. Если в ходе процесса не произошло никаких ошибок, все изменения автоматически заносятся в базу данных при его завершении.

Другим вариантом завершения транзакции является отмена изменений и возвращение содержимого БД в состояние, которое оно имело до начала процесса. Отмена изменений происходит при возникновении ошибок. Инициатором возникновения ошибки может быть система, либо ошибка может быть вызвана программным путем. Примером первого рода может служить попытка внести в базу данных клиента с номером (Customer No), уже присвоенным другому клиенту. Это вызовет ошибку и отмену изменений. Если данные не удовлетворяют необходимым критериям, вызвать ошибку и отмену изменений программно можно посредством функций TESTFIELD, ERROR и FIELDERROR.

Взаимосвязь процессов и транзакций

Транзакция всегда является частью процесса, тогда как процесс не обязательно включает в себя транзакции (например, обработка нажатия клавиш при вводе данных не вызывает блокирования каких-либо таблиц, то есть, не использует транзакций). Процесс может не использовать транзакций вообще, использовать одну или более транзакций. Для завершения транзакции перед началом следующей транзакции применяется функция COMMIT.

Процессы и версии

При запуске процесса с ним ассоциируется версия базы данных, являющаяся текущей на момент начала процесса. Таким образом, на протяжении всего процесса данные, с которыми работает процесс неизменны и фиксированы той версией БД, что ассоциирована с данным процессом. При этом текущая версия базы данных может изменяться вследствие активности других процессов.

Например, если в окне **General Ledger** вы запустили **Trial Balance Report** (генерацию текущего баланса), при создании балансового отчета будет использована одна полностью согласованная версия базы данных, в то время как другие пользователи могут в это же время вносить в БД изменения и добавлять новые данные.

Другим примером является создание резервной копии БД. На протяжении всего процесса записи копии базы данных используется одна версия БД. Таким образом, полученная копия самосогласованна. Принимая во внимание, что данные, образующие БД, могут быть распределены по нескольким физическим дискам, вы поймете, насколько важно фиксирование версии БД при резервном копировании.

Предположим, что во время процедуры резервного копирования другой пользователь инициировал обновление таблицы **General Ledger**. Пусть все данные, относящиеся к дебету, находятся в части базы данных, резервная копия с которой уже сделана, а данные кредитов – в части БД, которая еще не скопирована. В этом случае отсутствие механизма фиксирования версии базы данных при резервном копировании привело бы к получению несогласованной копии.

Как реализованы версии БД?

Каждая запись в БД имеет номер версии, ассоциированный с тем процессом, который создал или модифицировал данную запись. Если существует несколько версий данной записи – новым процессам на все время их работы будет предоставлена версия с наивысшим на момент начала процесса номером (текущая версия). В свою очередь во время работы этих процессов текущий номер версии может увеличиться.

Что происходит, если в момент использования одним пользователем какой-либо версии записи в БД другой пользователь обновляет эту запись?

При начале транзакции всегда создается новая версия БД. При этом старые версии не удаляются, а хранятся до тех пор, пока для этого есть свободное место в БД. Если процесс использует некоторую фиксированную версию БД – ему будут предоставлены записи, имеющие именно этот номер версии, даже если существуют более новые версии соответствующих записей.

Поэтому важно поддерживать в базе данных необходимое количество свободного места (по крайней мере 20%). Это «свободное» место на самом деле используется для хранения новых версий. Например, при записи данных в БД та версия, что была ассоциирована с процессом записи, должна существовать в базе до окончания процесса записи (на случай отмены изменений). Если в БД недостаточно места –

процесс записи не будет выполнен. Кроме того, существуют приоритеты выделения свободного пространства в БД для хранения версий. Если один пользователь запустил длительный процесс генерации отчета и ему необходимо место для хранения используемой версии БД, а в это же самое время другой пользователь начал запись данных в БД, то свободное место будет в первую очередь выделено процессу, ведущему запись. Если свободного места недостаточно для хранения обоих версий БД, то пользователь, генерирующий отчет, получит соответствующее сообщение об ошибке, и его процесс будет прерван. Таким образом, нехватка свободного места в БД может препятствовать нормальному завершению длительных транзакций чтения информации, например, созданию резервных копий БД.

Использование транзакций при записи информации и отмене изменений

Использование транзакций при обновлении содержимого БД Navision Server автоматизировано практически так же, как и управление версиями. Однако есть аспекты, на которые во избежание ошибок следует обращать внимание.

При обновлении БД системы Navision всегда имеет место транзакция. При этом создается новая версия всех обновляемых таблиц. Однако до завершения этой транзакции новая версия не предоставляется другим пользователям. Есть следующие два варианта завершения транзакции.

- Если транзакция прошла успешно, новые данные записываются в БД посредством явного или неявного вызова функции COMMIT. Это делает новую версию БД текущей, а место, занятое старой версией, освобождается.

- Если транзакция завершается ошибкой, производится отмена изменений. Текущая версия базы данных не изменяется, а дисковое пространство, выделенное для хранения новой версии БД, которая создана в начале неудачной транзакции, освобождается.

Транзакция начинается при вызове каким-либо процессом функции LOCKTABLE. Эта функция явным или неявным образом вызывается при обновлении таблицы. Начиная с этого момента и до своего завершения этот процесс использует новую, только что созданную версию таблицы. Все остальные таблицы, которые не были блокированы, по-прежнему используют старую версию, которая была текущей в момент начала процесса.

Таким образом, обеспечивается согласование данных при операциях записи, даже если другие пользователи производят изменения в БД одновременно с вами. При блокировании таблицы создается новая версия этой таблицы. Этот новый номер версии присваивается всем добавляемым или изменяемым записям таблицы. Другие пользователи не имеют доступа к этой новой версии и не могут блокировать таблицу.

Все остальные пользователи, работающие в это время будут использовать БД текущей версии или той версии, которая была текущей в момент начала их процессов. Другие пользователи, пытающиеся обновить эту же таблицу, будут ждать освобождения таблицы, после чего блокируют ее и создадут новую версию. При этом они получат версию БД, содержащую все изменения, внесенные всеми другими пользователями до настоящего момента, в частности, и с момента начала процесса до момента блокирования таблицы. До этого момента единственные данные, которые они могли получить об изменяемой таблице, – это номер версии, который был текущим в момент начала процесса.

Поэтому очень важно явным образом блокировать таблицу (функцией LOCKTABLE) перед тем, как считывать из нее какие-либо записи (функциями GET или FIND), если на ее данных будут базироваться дальнейшие обновления.

Пример

Вы считываете последнюю запись в таблице **Ledger**, чтобы узнать номер последней строки в таблице. Он необходим вам, чтобы назначить номер записи, которую вы собираетесь добавить. Этот номер должен быть на единицу больше номера последней имеющейся в таблице записи. После этого вы вызываете функцию INSERT, чтобы добавить новую запись. На первый взгляд, все правильно.

Однако предположим, что в то время, как вы начали свои действия, другой пользователь уже в течение 5 минут обновлял таблицу **General Journal**. Ни вы, ни созданный вами процесс не знают об этом. Ваш процесс получил информацию, которая была текущей 4 минуты назад. При чтении таблицы вы определили номер последней записи в старой версии таблицы (той, которая была текущей на момент начала ее обновления другим пользователем, так как никто другой также не мог блокировать таблицу и создать новых версий после того, как она уже была блокирована).

Тогда при вызове функции INSERT будет выполнен неявный вызов функции LOCKTABLE, и ваш процесс будет ожидать освобождения таблицы. При завершении другой процесс обновит таблицу, и именно она станет текущей с этого момента. Вы блокируете таблицу и получите доступ к обновленной таблице. В ней добавлено 2 000 новых записей, и тот номер записи, которую вы пытаетесь добавить, будет занят. Вы получите сообщение об ошибке.

Чтобы избежать подобной ситуации, вам следовало бы явным образом вызвать функцию LOCKTABLE перед операцией чтения из таблицы. В этом случае ваш процесс ожидал бы освобождения таблицы другими процессами. После этого вы получили бы последнюю обновленную версию таблицы. Вы определили бы номер последней записи таблицы в самой последней версии БД – той, которую вы в настоящий момент создаете. После этого можно добавить новую запись с номером, на единицу большим того, что был считан из таблицы. Таким образом, блокирование таблицы перед ее чтением гарантирует вам получение последней внесенной информации и отсутствие конфликтов, связанных с работой других пользователей.

Оптимистичный параллелизм

Что такое оптимистичный параллелизм?

Оптимистичный параллелизм (optimistic concurrency) означает, что запись не блокируется перед тем, как вы считываете ее для последующего обновления. В то время как вы просматриваете запись, она не блокирована; блокирование производится непосредственно перед началом записи обновленных данных. Если вы не обновляете запись, она не блокируется. Если вы обновляете запись и никто не изменяет ее одновременно с вами, операция пройдет корректно. И только в том случае, если кто-то обновил запись в промежутке между тем, как вы считали ее, и моментом, когда вы заносите обновленную запись в БД, возникнут ошибки. Поэтому такого рода параллелизм был назван «оптимистичным» – в силу того, что он оптимистично подразумевает отсутствие конфликтов из-за одновременной работы пользователей с одними и теми же данными.

Можно ли утверждать, что имеет место оптимистичный параллелизм при использовании версионной БД, использующей обновления на базе транзакций и блокирующей всю таблицу (а не только отдельную запись) при каждом обновлении записи? Бесспорно, для реализации оптимистичного параллелизма код должен быть написан специальным

образом. Однако при внесении пользователями изменений в БД с использованием форм оптимистичный параллелизм реализуется автоматически. А это наиболее распространенный способ обновления базы данных.

Реализация оптимистичного параллелизма

Встречаются ситуации, когда от момента чтения записи до момента ее обновления проходит значительное время, и вы не хотите держать таблицу блокированной все это время. В таком случае вы можете действовать следующим образом. Сначала считать необходимые записи, не блокируя таблицу. Потом, когда вы готовы обновить таблицу, вы блокируете ее с помощью явного вызова функции LOCKTABLE, установив при этом значение *TRUE* в параметре VersionCheck. Теперь можно обновить таблицу, считывая и записывая любые ее записи.

Если эти записи не были изменены каким-либо другим процессом, операция будет выполнена. В противном случае операция завершится ошибкой, и вы получите сообщение, извещающее, что запись была обновлена кем-то еще. Это та же самая ошибка, которая возникает, если вы изменяете запись, а другой пользователь одновременно обновляет ее.

Как работает оптимистичный параллелизм?

Как было сказано выше, при изменении записи ей присваивается номер версии, который ассоциирован с процессом, выполняющим эту операцию. После вызова функции LOCKTABLE с установленным параметром VersionCheck, каждый раз при обновлении записи производится проверка ее текущего номера версии. Если он по-прежнему меньше или равен номеру версии, ассоциированному с данным процессом, это означает, что запись не была изменена другими процессами и что можно продолжить ее обновление. Если же текущая версия записи имеет более высокий номер версии, чем номер версии, связанный с нашим процессом, это означает, что данная запись была изменена другим процессом после начала нашего процесса. В этом случае происходит ошибка.

При любом изменении содержимого БД перед обновлением записей неявно вызывается функция LOCKTABLE с установленным параметром VersionCheck. Таким образом, можно утверждать, что в большинстве практических ситуаций, возникающих при работе пользователей, система Navision использует оптимистичный параллелизм.

В качестве примера программного кода, реализующего оптимистичный параллелизм, можно рассмотреть любую из процедур записи журнала после длительной обработки данных. В ходе процесса записи мы считываем таблицу журнала, чтобы получить данные для обновления. При этом мы не блокируем ее, чтобы другие пользователи могли продолжать добавлять записи в шаблоны журнала и командные пакеты (batches). Когда мы готовы обновить или удалить запись из журнала, мы вызываем функцию LOCKTABLE с установленным параметром VersionCheck. Если другие пользователи не изменяли данный журнал или командный пакет, процедура записи будет выполнена. В противном случае возникнет ошибка, и будет выполнена отмена изменений, как описано выше.

Встроенные функции создания резервной копии и восстановления БД по резервной копии

Подобно многим другим базам данных, стандартная БД Navision имеет средства, позволяющие создать ее резервную копию и восстановить содержимое БД по резервной копии.

Как было сказано выше, вы можете создавать резервную копию БД в то время, как другие пользователи работают с базой данных. Однако создание резервной копии потребует существенно больше времени, если другие пользователи одновременно вносят изменения в базу данных.

При создании резервной копии порождается файл, который является целостной копией всей базы данных на определенный момент времени. В резервной копии сохраняются только данные. Если вы снова обратитесь к диаграмме, приведенной в разделе 3.1, вы увидите данные, сохраняемые в резервной копии – объекты приложений, общие данные для всех компаний и табличные данные каждой компании.

Обязательно ли делать резервную копию всех данных?

Нет, не обязательно. В резервную копию можно включить лишь определенные объекты, например, объекты приложений. Можно включить в копию любую комбинацию объектов, например, табличные данные компании A, либо табличные данные компании A и объекты приложений, либо табличные данные обеих компаний и общие данные всех компаний. Однако невозможно провести более детальное разделение данных на размещаемые и неразмещаемые в резервную копию. Например, нельзя сделать резервную копию лишь нескольких

таблиц одной из компаний. Вам придется либо создать резервную копию всех данных компании, либо отказаться от резервной копии.

Восстановление данных из резервной копии

Подобно тому, как вы можете выбрать объекты, подлежащие включению в резервную копию, вы можете выбрать и объекты, которые должны быть восстановлены в БД из резервной копии. Однако существуют некоторые ограничения. Следует принимать во внимание следующие соображения.

1 Процесс восстановления данных организован таким образом, что никакие данные не могут быть потеряны при восстановлении из резервной копии. Поэтому вы не можете восстановить какие-либо объекты в БД из резервной копии прямо поверх имеющихся в БД аналогичных объектов.

2 Если вам необходимо восстановить всю базу данных из полной резервной копии, вам нужно создать пустую базу данных (щелкните **File, Database, New**) достаточного размера.

3 Если вам необходимо восстановить из резервной копии только объекты приложений или данные, общие для всех компаний, вам нужно создать пустую базу данных (щелкните **File, Database, New**) достаточного размера. Невозможно восстановить данные, общие для всех компаний, в базу данных, где уже есть какие-либо таблицы.

4 Если вам необходимо восстановить данные конкретной компании, следует помнить о двух вещах.

Во-первых, в базе данных не должно быть данных компании с тем же именем, что и компания, данные которой вы восстанавливаете из резервной копии. Программа восстановления не заменяет данные существующих в БД компаний. При необходимости следует переименовать существующую компанию (щелкните **File, Company, Rename**). После этого можно восстановить данные из резервной копии. Рекомендуется документировать подобные действия.

Во-вторых, определения таблиц, записанные в таблице **Object**, должны соответствовать формату таблиц, которые вы восстанавливаете из резервной копии. Если какие либо типы данных в полях таблиц различаются – программа восстановления не позволит вам импортировать эту таблицу. Если программа

восстановления не может импортировать данные в таблицу, она импортирует их в копию соответствующей таблицы. Этой новой таблице присваивается новый номер (со значением около одного миллиарда). К сожалению, клиентская лицензия не позволяет получить доступ к таблицам с такими номерами. Чтобы использовать таблицы с такими номерами, необходима лицензия разработчика.

5 Процесс восстановления ни в коем случае не должен производиться с клиентского рабочего места, используя удаленный доступ к серверу. Восстановление следует проводить непосредственно на сервере при работе в автономном режиме. В противоположность процессу создания резервной копии сервер не может создать отдельную версию БД для записи восстанавливаемых данных.

Процедура создания резервной копии и восстановления БД из резервной копии

Команды создания резервной копии БД и ее восстановления из резервной копии находятся в меню **Tools**. В конфигурации по умолчанию эти команды доступны всем пользователям, если в БД не установлены иные права доступа. Рекомендуется разрешить создание резервных копий и восстановление данных в БД из резервных копий только определенным пользователям, так как такая возможность представляет угрозу целостности и конфиденциальности информации.

Файл резервной копии может содержать таблицы прав доступа, однако они не используются при восстановлении БД из резервной копии. После окончания процесса восстановления пользователь, производивший этот процесс, имеет системные привилегии (super user) по отношению к базе данных и имеет неограниченный доступ к ее содержимому. Поэтому возможность резервного копирования и восстановления БД следует предоставлять только авторизованным сотрудникам. Как только пользователь, производивший восстановление БД, завершает свою сессию и начинает новую, система контроля прав доступа вступает в работу. Кроме того, рекомендуется ограничить доступ к файлам, содержащим резервные копии БД, так как любой, кто имеет доступ к клиентской программе, может создать новую БД и восстановить в нее содержимое основной БД из резервной копии.

Можно ли просто сохранять файлы БД на магнитной ленте?

Вы можете использовать систему хранения данных на магнитных лентах, а также любые стандартные средства создания резервных копий, однако при этом следует быть очень осторожным. Существует опасность создать резервную копию БД в то время, когда с базой данных работают пользователи. Такая копия будет некорректной. Это одна из наиболее частых причин повреждения БД.

Создание резервной копии файла БД

При создании резервной копии файлов БД без использования стандартных средств Navision, рекомендуется следующая последовательность действий.

1. Перед созданием резервной копии завершите работу стандартного сервера.
2. Соединитесь с БД в автономном режиме, используя клиентскую программу, установленную на сервере. Выполните тест базы данных (рекомендуется провести хотя бы минимальное тестирование).
3. Завершите клиентскую сессию и создайте резервную копию файлов БД.
4. После создания резервной копии снова запустите стандартный сервер.

Такая процедура позволяет гарантировать, что при создании резервной копии файлы БД не будут повреждены.

Процедура тестирования БД – проверка содержимого БД

Во избежание повреждения данных рекомендуется регулярно проверять целостность БД и согласованность содержащихся в ней данных. Эта процедура может быть выполнена с любого клиентского рабочего места.

Без запуска интерактивной сессии клиентской программы

Тестирование БД можно выполнить без открытия интерактивной сессии клиентской программы. Следует запустить программу из командной строки, установив параметр DBTest.

Пример:

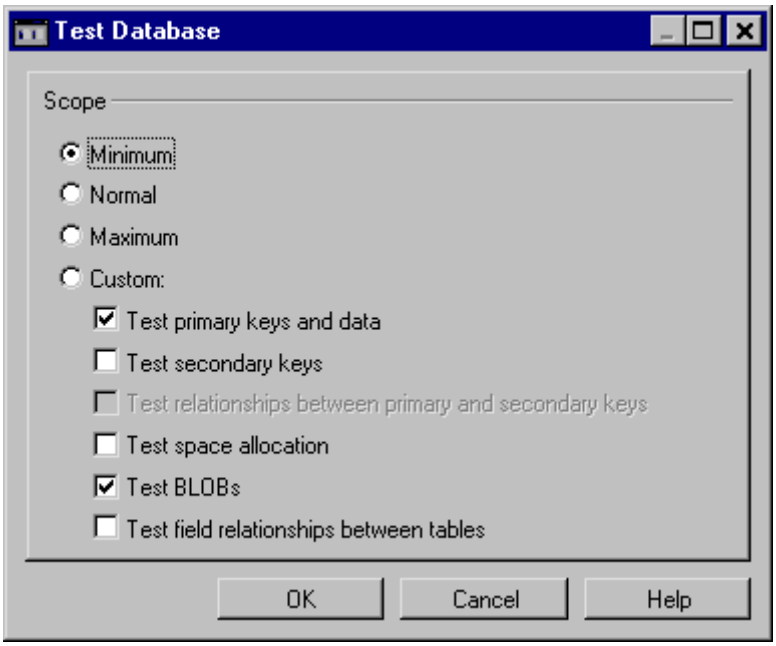
```
FIN servername=Server1,dbtest=normal
```

Допустимые значения параметра DBTest таковы: min, max и normal. Более детальная настройка параметров тестирования БД доступна при запуске теста из интерактивной сессии Navision.

Из интерактивной сессии клиентской программы

Чтобы начать тестирование, щелкните **File, Database, Test**.

Вы увидите окно **Test Database**.



Процедура тестирования устанавливается выбором одной из четырех предложенных опций (например, **Minimum**). Выбор какой-либо опции подразумевает выполнение лишь тех тестов, которые предусмотрены в данном варианте тестирования.

Приведенная ниже таблица описывает различные опции тестирования.

Опция	Проверяемые поля	Что проверяется
Minimum	Проверяются основные ключи и данные	Могут быть считаны все записи всех таблиц. Записи отсортированы в порядке возрастания по основному ключу. Тип содержимого всех полей соответствует типу данного поля.
	Проверяются BLOB-	Все BLOB-поля (например, поля для хранения фотографий товаров) могут

	поля	быть считаны.
Normal	Аналогично опции Minimum, кроме того:	
	проверяются вторичные ключи	Все вторичные ключи всех таблиц могут быть прочитаны. Сортировка записей выполнена корректно по вторичным ключам. Все поля при сортировке имеют соответствующий тип поля.
	проверяется свободное пространство	Все доступное пространство в БД либо свободно, либо занято ключами.
Maximum	Аналогично опции Normal, кроме того:	
	проверяются взаимосвязи	Верна связь между первичными и вторичными ключами.
	проверяются взаимосвязи полей	Все поля, связанные с другими полями, доступны из связанных с ними полей
Custom	По умолчанию выполняются те же проверки, что и для опции Maximum	По выбору.

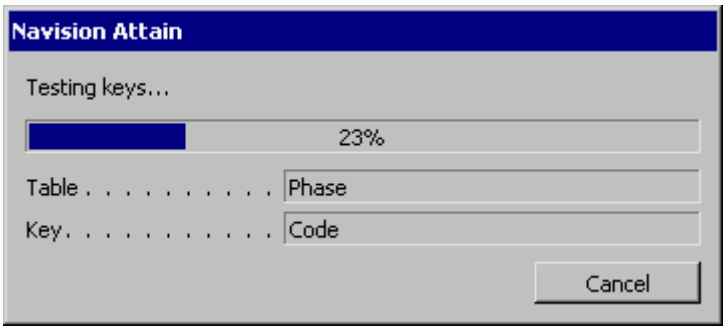
Первичные и вторичные ключи, упомянутые в приведенной выше таблице, иногда называют индексами. Они используются, в частности, для сортировки информации. Ключи определяют способ упорядочивания информации в таблице. Например, можно отсортировать данные о клиентах по именам клиентов или по их номерам.

Необходимая частота тестирования БД зависит от надежности остальных частей системы (в частности, сети) и ваших требований к надежности хранения информации в БД. Рекомендуется всегда тестировать БД перед созданием резервной копии, в особенности если для ее создания вы используете не стандартные средства Navision, а утилиты резервного копирования сторонних производителей.

Примечание

Проверка свободного пространства в БД не может быть выполнена, если с системой работает более одного пользователя. Если при запуске теста, включающего в себя проверку свободного пространства в БД, с системой работают другие пользователи, данная проверка будет пропущена, однако все остальные проверки будут выполнены обычным образом.

Во время тестов выводится показанное ниже окно состояния.



В случае возникновения ошибки программа прерывает свою работу и выводит сообщение, поясняющее тип возникшей ошибки и операцию, которая вызвала ошибку. В этом случае нужно предпринять следующие шаги.

1. Создать резервную копию БД с помощью встроенной утилиты резервного копирования Navision.
2. Создать новую БД и восстановить ее содержимое из резервной копии.
3. Выполнить тестирование новой БД с опцией Maximum.
4. Если тестирование не выявило ошибок, проверьте правильность данных о компаниях.
5. После этого можно продолжить работу с новой базой данных.
6. Если вышеописанные действия не помогли, свяжитесь со службой поддержки.

Внимание

Не удаляйте поврежденную базу данных (ту, которая содержит ошибку) до тех пор, пока новая база данных не проработала без ошибок некоторый период времени.

Не забывайте своевременно создавать резервные копии и проверять, что БД может быть восстановлена из этих копий.

Использование кэширования для ускорения доступа к БД

Клиентская и серверная программы имеют несколько опций, правильная установка которых может повысить производительность системы за счет экономии времени, необходимого для передачи данных от БД клиентам. Все эти опции связаны с кэшированием. В этом разделе мы рассмотрим все три типа кэшей, используемых Navision.

Кэш-память

Кэш-памятью (cache) называется область оперативной памяти, которая резервируется и используется СУБД для временного хранения данных. Общий размер кэш-памяти не может превышать 1 Гигабайта (Гбайт). Кэширование существенно ускоряет доступ к данным за счет размещения части записей БД в оперативной памяти компьютера. Чтение из оперативной памяти требует намного меньше времени, чем чтение тех же данных с диска.

Вся выделенная на сервере Navision кэш-память разделяется на две области: кэш чтения (для кэширования транзакций чтения из БД) и кэш записи (для кэширования транзакций записи). Navision устанавливает размер обоих областей автоматически. Для кэша чтения отводится весь доступный объем памяти, но не менее 1/3 общего объема кэш-памяти.

При выборе объема резервируемой кэш-памяти следует руководствоваться двумя правилами.

1. Чем больше объем кэш-памяти, тем более эффективным будет кэширование. Разумеется, нет смысла выделять кэш-память, объемом превосходящую размер вашей БД.
2. Объем выделенной кэш-памяти и количество памяти, необходимое для нормального функционирования ОС в сумме не должны превышать полного объема установленной в компьютере оперативной памяти. В противном случае ваша ОС использует свопирование кэш-памяти на диск, что замедлит работу системы.

Общее правило для выбора объема кэш-памяти таково – стандартный сервер должен иметь около 2 Мбайт оперативной памяти на каждую клиентскую сессию Navision и, кроме того, количество памяти, необходимое для работы ОС и других приложений.

Кэширование записи

Кэширование записи (commit cache) повышает производительность системы, и, в частности, позволяет быстрее реагировать на действия пользователя за счет того, что сервер Navision осуществляет запись данных не на диск, а в оперативную память. Для кэширования транзакций записи выделяется от 1/3 до 2/3 от общего объема кэш-памяти. Когда объем кэшированных, но не записанных на диск данных достигает 2/3 от общего объема кэш-памяти, операциям записи присваивается более высокий приоритет, чем операциям чтения из БД, и, таким образом, происходит разгрузка кэша записи. Запись кэшированных данных на диск выполняется также в промежутки времени, когда отсутствуют транзакции чтения из БД.

Кэш объектов

В отличие от двух описанных выше кэшей, кэш объектов (object cache) создается на клиентской рабочей станции. Он применяется для хранения используемых данным клиентом объектов, что позволяет выполнить загрузку объектов из таблицы **Object** в БД лишь один раз, а при необходимости повторного использования объектов загружать их из оперативной памяти локального компьютера. Чем больше различных операций выполняет данный клиент, тем большее число объектов требуется для их выполнения и тем больше должен быть объем памяти, выделенной для их кэширования.

Имеется ряд причин, препятствующих разработке программного кода на работающих системах клиент-сервер. Одной из этих причин является кэширование объектов. Кэш объектов должен обеспечить клиентской программе возможность запрашивать каждый объект (или, хотя бы, большинство объектов) из БД лишь один раз, а при повторной необходимости извлекать данный объект из локальной оперативной памяти. Однако если разработчик изменил объект в БД уже после того, как данный объект был считан клиентом и сохранен в его кэш-памяти, клиент (даже если это другой разработчик) не узнает об изменении объекта и будет продолжать использовать его старую копию. Таким образом, может возникнуть ситуация, когда разные пользователи Navision используют различные версии одного и того же объекта, что повлечет за собой множество проблем.

Более того, может оказаться, что два или более разработчиков одновременно работали над одним и тем же объектом. В этом случае в БД останется последний из записанных объектов, а все варианты того же объекта, записанные ранее, будут потеряны.

Важность кэширования

Для повышения производительности системы рекомендуется использовать кэширование. Кэшем называют некоторый объем оперативной памяти, используемый для временного хранения той части БД, которая наиболее активно используется в настоящий момент. Если запрашиваемая СУБД информация имеется к кэш-памяти, системе не приходится выполнять медленную операцию доступа к диску, что значительно повышает общую производительность. Кэш СУБД в любой момент времени содержит информацию, которая недавно использовалась одним из клиентов. Давно не используемая информация в кэш-памяти автоматически заменяется более свежей. Размер кэш-памяти существенно влияет на общую производительность системы.

Технология суммирования индексных полей (SumIndexField Technology, SIFT)

Технология суммирования индексных полей (SIFT) является другим важным фактором, обеспечивающим высокую производительность стандартной БД Navision. Эта технология обеспечивает два основных преимущества: повышенную производительность при извлечении данных и онлайновую аналитическую обработку (OLAP, On Line Analytical Processing), которая позволяет использовать развитые алгоритмы отбора информации и извлекать из БД необходимую вам информацию. Технология SIFT предоставляет еще одно преимущество – все балансы в Navision могут быть вычислены, то есть нет необходимости хранить их в базе данных. Поэтому не требуется обновлять главные записи, например, в таблице **Customer**, при записи данных. Это сокращает время, необходимое на запись информации в БД и способствует эффективному использованию оптимистичного параллелизма.

Технология SIFT реализована с использованием вычисляемых полей (FlowFields), фильтров вычисляемых полей (FlowFilters) и полей сумм индексных полей (SumIndexFields). Концепции их использования рассматриваются подробнее в следующих частях курса. В данном разделе мы лишь дадим базовую информацию об этих полях и их использовании.

Вычисляемые поля

Вычисляемые поля позволяют СУБД рассчитывать значение некоторых полей, используя другие поля БД в качестве исходных данных. Существует несколько типов вычисляемых полей.

Вычисляемое поле может содержать:

- сумму или среднее значение количественных полей в другой таблице, например, поле **Balance** в таблице **Customer;**
- максимальное или минимальное значение в столбце другой таблицы;
- количество записей в другой таблице;
- флаг, значение которого зависит от того, существует ли некоторая запись в другой таблице;
- значение заданного поля в другой таблице.

Поскольку значения вычисляемых полей рассчитываются на основе других полей в БД, для их хранения не требуется место в той таблице, где они определены. Значение этих полей вычисляется только тогда, когда клиентская программа запрашивает их.

Поля сумм индексных полей

В отличие от вычисляемых полей, значения полей сумм индексных полей (SumIndexFields) хранятся в БД и требуют места. Обычно поле SumIndexField используется, когда требуется вычислить сумму или среднее значение столбца в таблице. Например, поле **Amount** в таблице **Cust. Ledger Entry** является полем SumIndexField. Клиентская программа в любой момент может запросить сумму некоторого набора записей таблицы по данному полю. Такой запрос может быть реализован в коде программы или быть выполнен косвенно, с использованием вычисляемого поля.

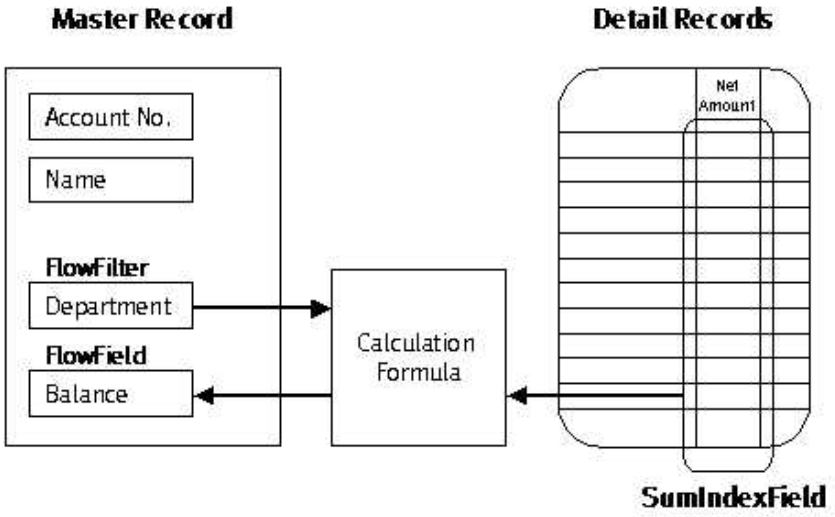
Поля SumIndexFields являются частью ключей стандартной базы данных и существенно увеличивают долю пространства в БД, занимаемую ключами. Следует помнить об этом при добавлении к ключам полей SumIndexField.

Фильтры вычисляемых полей

Как следует из названия, фильтры вычисляемых полей тесно связаны с самими вычисляемыми полями. Они также являются виртуальными, т.е. не требуют места для своего хранения. Поля фильтров предоставляют возможность установить критерии отбора, применяемые к полям, которые используются в качестве исходных данных каким-либо вычисляемым полем. Функция полей фильтров состоит в применении заданных фильтров к исходным полям вычисляемых полей. Таким образом, можно динамически изменять формулу, используемую вычисляемым полем непосредственно при

выполнении программы. Это обеспечивает повышенную гибкость технологии SIFT.

Приведенная ниже функциональная диаграмма иллюстрирует взаимосвязь и функционирование вычисляемых полей, фильтров и полей SumIndexField при вычислении затребованной клиентом суммы.



Создание резервных копий на сервере

С системой Navision поставляется утилита резервного копирования БД на сервере HotCopy. HotCopy устанавливается вместе с Navision Server и располагается в том же каталоге, что и Navision Server.

Утилита HotCopy предназначена для создания резервной копии БД без остановки сервера, то есть без прерывания обслуживания клиентов. Когда HotCopy приступает к копированию базы данных, она создает в заданном каталоге файл базы данных с тем же именем, что и копируемая база данных. Однако у его имени используется расширение `.SAV`. Таким образом, если ваша БД называется `Production.fdb`, имя резервной копии будет `Production.fdb.SAV`. Это имя используется на протяжении процесса копирования информации. После его завершения файл переименовывается так, что его имя совпадает с именем исходной базы данных - `Production.fdb`. Для предотвращения уничтожения исходной БД программа контролирует заданный для размещения копии каталог. Если это тот же каталог, где расположена исходная БД, резервной копии будет присвоено имя `Production.fdb.BAK`.

Утилита HotCopy работает аналогично средствам резервного копирования клиентской программы Navision, то есть позволяет создавать резервную копию, не прерывая работы пользователей с базой данных. Поскольку HotCopy запускается на сервере, создание резервной копии требует существенно меньше времени, чем аналогичный процесс, выполняемый на клиентском рабочем месте.

Внимание

Резервная копия, созданная с помощью утилиты HotCopy, представляет собой согласованную копию БД в фиксированный момент времени (начало процесса резервного копирования). Она не содержит информации о транзакциях, совершенных после запуска процесса копирования. Navision не ведет журнала транзакций. Если вам требуется такая возможность, вам следует перейти от стандартной СУБД Navision к использованию СУБД Microsoft SQL Server.

Утилита HotCopy не имеет графического интерфейса. Запустить HotCopy можно несколькими способами:

- запустить HotCopy из командной строки;
- создать ярлык и использовать его для запуска;
- создать командный файл, запускающий HotCopy.

Перед тем, как привести примеры использования HotCopy, мы рассмотрим различные параметры этой программы.

Параметр	**Пояснение**	**Обязательный / Необязательный**
source	Файл БД, для которой создается резервная копия. Этот параметр может задавать либо имя БД, либо имя файла, содержащего БД. Если требуется выполнить резервное копирование нескольких файлов БД, следует задать имена всех файлов. При использовании конфигурационного файла создания резервной копии в	Обязательный

Параметр	Пояснение	Обязатель-ный / Необяза-тельный
	качестве параметра source следует указать имя конфигурационного файла. Можно использовать абсолютный или относительный путь к файлам БД. Если путь не задан, предполагается, что файлы БД находятся в том же каталоге, где хранится утилита `HotCopy.exe`. Можно также указать полный путь к конфигурационному файлу. Если в качестве параметра source задан конфигурационный файл, все остальные параметры, заданные в командной строке, игнорируются. Формат конфигурационного файла рассматривается ниже.	
destination	Путь к каталогу, где должна быть размещена резервная копия. Если указано несколько исходных файлов ДБ для резервного копирования и задан только один путь для размещения резервной копии, все файлы будут сохранены в указанном каталоге. Если все файлы БД имеют одинаковые имена, следует задавать различные каталоги для размещения резервных копий. Файлы копируемых БД будут созданы в указанном каталоге с именами, соответствующими исходным файлам. Если в указанном каталоге уже имеется файл с таким именем (например, предыдущая копия БД), он будет уничтожен. HotCopy не сообщает о том, что она переписывает файл с заданным именем.	Обязателен при вызове из командной строки. Не должен применяться при использовании конфигурационного файла

Параметр	Пояснение	Обязательный / Необязательный
	Если каталог для размещения резервной копии совпадает с каталогом, где хранится копируемая БД, файлу резервной копии БД будет добавлено расширение "`.bak`". Например, копия БД `database.fdb` будет сохранена в том же каталоге с именем `database.fdb.bak`. Если в данном каталоге уже имеется предыдущая копия с тем же именем, она будет заменена новой. HotCopy не сообщает о том, что она переписывает файл с заданным именем.	
dbtest	При задании этого параметра перед созданием резервной копии выполняется тестирование БД. Можно выбрать один из трех стандартных вариантов тестирования, соответствующих тестам БД, которые доступны из интерактивной сессии Navision. Допустимые значения этого параметра: minimum, normal или maximum. Если сервер Navision работает на компьютере под управлением ОС UNIX, при полном тестировании БД не будут проверятся взаимосвязи таблиц.	Необязательный
cc	Задание данного параметра означает проведение проверки резервной копии по окончании копирования. Проверка состоит в поразрядном сравнении резервной копии с исходной БД. Допустимые значения: yes или no.	Необязательный
description	Значением этого параметра является	Необяза-

Параметр	Пояснение	Обязатель-ный / Необяза-тельный
	текстовая строка, описывающая резервную копию. Эта строка используется в поле "source" в протоколе работы приложения (event log) и отображается при его просмотре с помощью Event Viewer. Также она используется в поле Subject в сообщении электронной почты, которое может быть отправлено администратору системы.	тельный
email	Задает адрес электронной почты, по которому следует отправить сообщение, информирующее об успешности резервного копирования. Сообщение содержит следующую информацию: Subject: Резервная копия в понедельник (краткое описание копии). Резервное копирование начато 2001/03/15 10:02:50. Резервное копирование успешно завершено 2001/03/15 10:06:06. В случае неудачного завершения процесса резервного копирования сообщение будет содержать только краткое описание копии и информацию об ошибке. При использовании ОС UNIX для отправки сообщений электронной почты на сервере должна быть установлена программа sendmail.	Необяза-тельный
servername	Имя сервера Navision Server, к которому должна подключится утилита HotCopy при запуске процедуры	Необяза-тельный

Параметр	Пояснение	Обязательный / Необязательный
	резервного копирования. Если данный параметр не задан, HotCopy попытается установить соединение с сервером, работающим на компьютере с именем "localhost". Если это также не удалось из-за того, что не запущен Navision Server, HotCopy использует локальное соединение с файлом БД. Иными словами для создания резервной копии не требуется работающий сервер стандартной БД. Рекомендуется использовать утилиту HotCopy при работающем сервере Navision Server.	
osauthentication	Задание этого параметра вызывает проверку авторизации пользователя, запустившего процедуру резервного копирования для доступа к БД. При этом используются его имя и пароль Windows. Единственное допустимое значение: *yes*. Можно использовать либо этот параметр, либо параметры ’user’ и ’password’.	Необязательный
user	Если не используется авторизация Windows NT, этот параметр задает имя пользователя, выполняющего резервное копирование.	Необязательный
password	Если не используется авторизация Windows NT, этот параметр задает пароль пользователя, выполняющего резервное копирование.	Необязательный

Параметр	Пояснение	Обязательный / Необязательный
nettype	Сетевой протокол, используемый клиентами для соединения с сервером. Допустимые значения: netb (NetBIOS) и tcp (TCP/IP)	Необязательный

При вызове HotCopy из командной строки с параметрами или с использованием конфигурационного файла можно задать одновременно любое число параметров.

Запуск HotCopy из командной строки

Приведенный ниже пример иллюстрирует использование обязательных параметров.

```
HotCopy source="C:\Attain\databasename.fdb"
destination="C:\Backups"
```

Если не указан полный путь к файлам БД, HotCopy подразумевает, что БД расположена в том же каталоге:

```
HotCopy source=database.fdb destination="D:\Database
Backups\Attain"
```

В приведенных примерах в корневой каталог диска C будет помещена резервная копия БД:

```
HotCopy source=database.fdb destination=C:\
```

или

```
HotCopy source="C:\Attain Databases\database.fdb"
destination=\
```

Эта команда создаст резервную копию в том же каталоге, где расположена исходная БД. Резервная копия будет названа database.`fdb.BAK`:

```
HotCopy source="C:\Attain Server\database.fdb"
destination="C:"
```

или

```
HotCopy source=”C:\Attain Server\database.fdb”
destination=””
```

Этот пример использует все параметры и иллюстрирует создание резервной копии БД, состоящей из двух файлов:

```
HotCopy source=C:\database\database1.fdb
c:\database\database2.fdb
destination=”C:\Databases\Backups”
”C:\Databases\Backups” description="This is the backup
for Monday" email=username@navision.com
servername=myserver user=myname password=mypassword
cc=yes nettype=tcp
```

Все параметры в приведенном выше примере разделены пробелами.

Запуск HotCopy с использованием конфигурационного файла

```
HotCopy source=backupdescription.txt
```

Конфигурационный файл утилиты HotCopy

Конфигурационный файл содержит информацию, которую утилита HotCopy использует при создании резервной копии.
Конфигурационный файл может иметь любое имя, но он обязательно должен содержать два раздела:[Backup Files] и [Options], каждый из которых содержит одну или более строк, не являющихся комментариями. В разделе [Backup Files] необходимо задать имя БД в левой части и каталог для размещения резервной копии -- в правой части. Нельзя указать имя файла резервной копии: может быть задан лишь каталог для его размещения. В конфигурационном файле допускаются комментарии. Комментарием считается любая текстовая строка, начинающаяся со знака ";".

При использовании утилиты HotCopy рекомендуется использовать конфигурационный файл. Ниже приводится пример конфигурационного файла:

```
[Backup Files]

;Внимание: Это обязательный раздел и значение
```

```
;C:\Navision Programs\WW\Attain3_1\Database
Server\ww31base.fdb=c:\Navision Related

;Если в используемой ниже строке поставить кавычки
"C:\Navision Backups", это вызовет ошибку

;и создаст запись в журнале работы приложения(event
log): Event ID: 109 - Warning

;То же самое произойдет при неверном задании любого из
параметров, например, в случае, если указанный каталог
для размещения резервной копии не существует

ww31base.fdb=c:\Navision Backups



[Options]

; Точка с запятой обозначает комментарии.

;НЕ ИСПОЛЬЗУЙТЕ destination= в этом файле. В противном
случае вы получите сообщение об ошибке:

;There is not enough memory available to execute this
function

;dynamic_cast<DestinationTokenString*><pTokenNode-
>m_pTokenString>.



description="Server Backup Test 1"

;Эта строка задает поле Subj в сообщении электронной
почты



servername=NAVISERVER

user=SUPER

;Опция password не использована, так как пользователь
SUPER не имеет пароля
```

```
;Это тестовый пользователь, созданный в учебных целях


email=YourName@YourCompany.com

cc=yes

osauthentication=no

nettype=tcp
```

Использование HotCopy на компьютерах под управлением ОС UNIX

На компьютерах под управлением ОС UNIX параметр source следует задавать следующим образом:

```
Source=/database/database1.fdb /database/database2.fdb
```

Статус завершения процедуры резервного копирования заносится в лог-файл компьютера, на котором выполнялось копирование. На компьютерах под управлением ОС UNIX эта информация заносится в файл `hotcopy.log`.

Что следует принимать во внимание при использовании утилиты HotCopy

1 HotCopy может работать только на сервере и может создавать резервные копии только на локальных дисках. Это означает, что резервная копия не может быть размещена на разделяемом диске, физически подключенном к другому компьютеру, а также, что вы не можете задавать сетевой путь.

   Тем не менее, HotCopy может быть запущена на клиентской рабочей станции при условии, что клиент имеет доступ по чтению к базе данных. В этом случае файлы должны быть заданы полным сетевым именем. Можно также использовать разделяемые сетевые диски и сетевые пути при условии, что пользователь, запустивший HotCopy, имеет право записи на эти диски.

2 Файл резервной копии не подвергается сжатию. Поэтому для резервной копии требуется столько же дискового пространства, сколько и для копируемой БД.

3 HotCopy не может делать накопительные (инкрементальные) или разностные (дифференциальные) резервные копии.

4 HotCopy не имеет графического интерфейса для облегчения настроек. Поэтому есть два способа запуска HotCopy – с заданием параметров source и destination, либо с использованием конфигурационного файла.

5 Каждый параметр, содержащий пробел, должен быть заключен в кавычки.

6 Необходимо задавать путь к каждому из файлов БД, если только все они не расположены в том же каталоге, что и HotCopy.

7 При использовании конфигурационного файла число параметров source должно соответствовать числу параметров destination. При задании параметров в командной строке можно указать несколько файлов БД и один каталог для резервной копии – все файлы копии будут помещены в указанный каталог.

8 При попытке использовать сетевой диск или сетевое имя файла вы получите сообщение об ошибке: “От сервера получено следующее сообщение. Операционная система не смогла получить доступ к файлу. Пожалуйста проверьте правильность задания файлов и параметров.”

9 Если вы забыли поместить в кавычки имя каталога для размещения резервной копии, и это имя содержит пробелы, или если различается число параметров source и destination (только в случае, если у вас нет доступа по чтению и записи к каталогу destibnation), вы получите следующее сообщение об ошибке: “Число файлов в разделах 1 превышает число заданных укаталогов для размещения резервных копий 2.”

Глава 4

Опция SQL Server

В этой главе пойдет речь об использовании СУБД SQL Server совместно с Navision.

Решение, реализуемое вами для заказчика, может базироваться либо на СУБД Navision, либо на СУБД Microsoft SQL Server.

Для полного понимания всех нюансов, связанных с использованием SQL Server, следует изучить учебный курс *Navision SQL Server*. Настоящий курс содержит лишь вводную информацию по применению этой СУБД.

Данная глава состоит из следующих разделов:

- Преимущества использования СУБД SQL Server;
- Как работает база данных в формате SQL Server.

4.1 Преимущества использования СУБД SQL Server

Обе СУБД имеют свои преимущества. Главным достоинством SQL Server является масштабируемость. Объем стандартной БД не может превышать 64 Гбайт. Для БД в формате SQL Server такого ограничения не существует. Если объем базы данных вашего заказчика уже достаточно велик и продолжает расти, этот вариант может оказаться оптимальным.

Среди других преимуществ можно выделить следующее:

- *доступность:* используя стандартный SQL ODBC-драйвер корпорации Microsoft, вы можете иметь доступ к данным, хранящимся в системе Navision, из многих внешних приложений. Также вы можете осуществлять взаимодействие с другими SQL-продуктами с помощью триггеров и хранимых процедур SQL.
- *манипулирование данными:* используя Database Transformation Services (DTS) и Bulk Copy Program (BCP), вы можете внести или удалить данные из базы данных посредством простого графического или командного интерфейса.
- *защита от сбоев:* за счет интеграции с операционными системами корпорации Microsoft (Windows NT и Windows 2000) сервер SQL Server поддерживает организацию кластеров. Это позволяет нескольким компьютерам обрабатывать данные, и если основной сервер выйдет из строя, резервный возьмет на себя его функции.
- *администрирование:* вы можете настроить SQL-сервер БД так, чтобы он автоматически выдавал предупреждения («Alerts») в случае возникновения определенных ситуаций. Например, если не хватает дискового пространства для хранения базы данных, резервное копирование может осуществляться на уровне сервера, а не на клиентском уровне.

Сложность администрирования является, пожалуй, единственным недостатком SQL Server. В отличие от стандартной базы данных Navision, база данных в формате SQL Server требует регулярного обслуживания и контроля, поэтому заказчику, вероятно, придется нанять администратора баз данных. В этом случае также потребуются большие затраты на приобретение аппаратного обеспечения, однако именно текущие затраты на содержание администратора — это тот фактор, который должен учитываться в первую очередь.

4.2 Как работает база данных в формате SQL Server

Этот раздел содержит краткое описание того, как работает база данных в формате SQL Server, а также описание некоторых операций.

Как создавать резервные копии, восстанавливать и тестировать базу данных в формате SQL Server

Для создания резервных копий и восстановления данных следует пользоваться штатными средствами SQL Server. Однако нужно отметить, что будут функционировать и стандартные процедуры резервного копирования и восстановления, входящие в состав клиентской программы Navision, хотя их использование в общем случае не рекомендуется. Например, вы можете создать резервную копию файла базы данных в формате Navision, а затем восстановить его, преобразовав в формат базы данных SQL Server. Тем же способом можно осуществить и обратное преобразование. Помните, что использование стандартных средств резервного копирования и восстановления создает потенциальную угрозу безопасности системы, поэтому следует ограничивать доступ пользователей к резервным копиям данных.

Рекомендуется использовать встроенные средства резервного копирования SQL, так как они работают быстрее, чем аналогичные средства Navision. Данные могут записываться непосредственно на магнитную ленту или другие устройства хранения, а также на несколько физических жестких дисков одновременно. Для базы данных в формате SQL существует четыре типа резервного копирования: создание полной копии базы данных (как в системе Navision), создание разностных (накопительных) копий, создание копии файла базы данных и создание копии журнала транзакций. Такая гибкость позволяет сконфигурировать систему резервного копирования с учетом ваших специфических требований.

SQL Server имеет собственные средства для тестирования базы данных, однако они не могут использоваться для проверки целостности самих данных. Поэтому мы рекомендуем регулярно проводить тестирование базы данных тем же способом, как при использовании стандартной базы данных. Перед созданием резервной копии с помощью средств, не входящих в систему Navision, например, встроенных средств SQL Server, проведите хотя бы минимальное тестирование базы данных. В набор тестов, осуществляемых клиентской программой, включены тесты, которые можно проводить с использованием ПО SQL Server. В этом состоит

единственное различие между средствами тестирования для стандартной базы данных и для базы данных в формате SQL Server.

Более подробная информация по администрированию Microsoft SQL Server содержится в документации корпорации Microsoft.

Поддерживает ли SQL Server систему управления версиями?

Нет, не поддерживает. Каждая запись базы данных в формате SQL Server снабжена временной меткой, но различные версии одной и той же записи не сохраняются. Временная метка позволяет узнать, когда было произведено обновление записи, и это дает возможность клиентской программе имитировать некоторые свойства системы управления версиями, но далеко не все.

Если, используя базу данных в формате SQL Server, вы хотите создать полностью согласованный финансовый отчет, необходимо перед началом чтения заблокировать все таблицы, которые вы будете считывать. Для этого при создании отчета надо задать тип транзакции «Snapshot». При этом количество заблокированных таблиц будет гораздо больше, чем в случае базы данных, имеющей систему управления версиями.

Оптимистичный параллелизм (Optimistic Concurrency)

SQL Server реализует оптимистичный параллелизм (optimistic concurrency, используя временные метки, которыми снабжена каждая запись.

Когда пользователь открывает форму и считывает запись, форма запоминает ее временную метку. Когда этот пользователь пытается сохранить данную запись после внесения изменений, форма сравнивает запомненную временную метку записи с меткой, хранящейся в базе данных. Если они совпадают, запись обновляется вместе со своей временной меткой. Если же они различны, пользователю выдается сообщение об ошибке, говорящее о том, что данная запись была обновлена другим пользователем и что изменения сохранены не будут. В этом и заключается принцип оптимистичного параллелизма.

В SQL Server оптимистичный параллелизм применяется автоматически. Для Navision Server в команде LOCKTABLE необходимо использовать параметр Version Check. Однако для SQL Server использование этого параметра недопустимо. Если вы просто

считываете запись и позднее обновляете ее, то никаких дополнительных действий не требуется. Но при работе с Navision в большинстве случаев мы считываем несколько записей сразу, затем через некоторое время считываем их вновь и обновляем. Хотя этот механизм хорошо работает для Navision Server, в случае использования SQL Server дополнительные операции чтения препятствуют эффективному применению оптимистичного параллелизма. Следовательно, чтобы осуществить это для SQL Server, необходимо заблокировать таблицу перед первой операцией чтения. Именно поэтому журналы SQL Server блокируются на время осуществления записи данных.

Транзакции в SQL Server

Транзакции (внесение и отмена изменений) обрабатываются в SQL Server иначе, так как там отсутствует система управления версиями. С точки зрения пользователя, SQL Server работает так же, как и стандартная СУБД Navision, однако принципы работы этих систем совершенно различны. SQL Server заносит все транзакции записи в журнальный файл. Когда изменения вносятся в базу данных, журнальный файл помечается. Если внесенные изменения отменяются, журнальный файл считывается и изменения удаляются из базы данных.

Это означает, что внесение изменений в базу данных в формате SQL Server занимает немного больше времени, чем при использовании Navision Server, так как SQL Server должен повторить транзакции для журнального файла.

В остальном возможности обеих систем, связанные с внесением и отменой изменений в базе данных, совершенно одинаковы. SQL Server и Navision Server обладают идентичными наборами команд и функций.

Механизм блокировки в SQL Server

В Navision Server применяется только один механизм блокировки — блокировка таблиц. В SQL Server чаще всего используется блокировка записей, которая может быть преобразована в блокировку таблиц, если SQL Server решит, что этот способ предпочтителен. С этим связаны три эффекта, которые необходимо учесть.

Во-первых, если вы считываете и редактируете различные записи в таблице (например, хотите связать какие-либо элементы между

собой), другие пользователи могут успешно делать это одновременно с вами. Конечно, команда LOCKTABLE должна быть применена, но несколько пользователей могут считывать и редактировать различные записи в одной и той же таблице в одно и то же время. Если команда LOCKTABLE действует, то для блокировки записи надо лишь считать ее. Обновления такого типа осуществляются в Navision относительно редко.

Во вторых, если вы добавляете записи в базу данных в формате SQL Server (это наиболее распространенная операция при использовании Navision), необходимо не только дать команду LOCKTABLE перед чтением записей, но и немедленно считать последнюю запись в таблице. Это позволит заблокировать общую запись (common record), что будет эквивалентно блокировке таблицы. Каждая из добавленных вами после этого записей будет являться последней записью, то есть последняя запись в таблице останется заблокированной, и доступ другим пользователям будет запрещен до тех пор, пока не завершится процесс обновления.

Наконец, когда вы составляете финансовый отчет, данные в используемых таблицах должны быть согласованы. В Navision Server, где применяется система управления версиями, это делается автоматически. Однако в случае SQL Server даже команда LOCKTABLE не дает нужного эффекта, так как при этом записи блокируются по мере чтения. Поэтому для создания согласованного финансового отчета необходимо задать тип транзакции «Snapshot» («моментальная фотография»), что не позволит другим пользователям блокировать записи в таблице во время генерации отчета. Задание этого типа транзакции никак не влияет на данный процесс при использовании Navision Server, так как версии сами являются «моментальными фотографиями». Следовательно, при создании отчетов, которые должны содержать согласованные данные, рекомендуется задание типа транзакции «Snapshot» независимо от того, какой формат базы данных вы используете.

Функция RECORDLEVELLOCKING

Вы наверняка обратили внимание, что в некоторых случаях, описанных выше, необходимо знать, какой тип базы данных используется. Для этого существует функция RECORDLEVELLOCKING. Если она возвращает значение TRUE, значит вы используете базу данных в формате SQL Server, где нет системы работы с версиями, где оптимистичный параллелизм реализован на основе временных меток и блокировка осуществляется

на уровне записей. Если эта функция возвращает FALSE, значит вы используете базу данных в формате Navision Server, где применяется система работы с версиями, где оптимистичный параллелизм реализован на основе определения номера версии и блокировка осуществляется на уровне таблиц.

Технология технология суммирования в индексных полях (SIFT, Sum Index Field Technology) в SQL Server

Технология SIFT применяется для баз данных в формате SQL Server так же, как и при использовании Navision Server. К сожалению, эта технология реализована не так эффективно при работе с SQL Server. В Navision Server SIFT работает быстро, так как поля SumIndexFields привязаны непосредственно к ключу. SQL Server не допускает такой реализации, поскольку редактирование значений в индексных файлах невозможно.

При использовании базы данных в формате SQL Server поля SumIndexFields должны храниться в отдельной таблице. Это означает, что запрос на суммирование представляет собой дополнительную директиву select, и обновление таблицы, в которой производится суммирование, также приводит к обновлению нескольких записей в таблице SIFT базы данных. Таким образом, реализация SIFT в базе данных в формате SQL Server связана с осуществлением множества дополнительных транзакций.

Однако с точки зрения пользователя SIFT работает одинаково в обоих вариантах. SQL Server и Navision Server обладают идентичными наборами команд.

Глава 5

Обеспечение безопасности в системе Navision

В этой главе мы познакомим вас с принципами обеспечения безопасности в системе Navision. Предполагается, что в процессе изучения данной главы вы будете использовать клиентское ПО Navision, работающее в автономном режиме.

Вы создадите учетную запись пользователя системы и проверите, как действуют ограничения прав доступа.

Эта глава состоит из следующих разделов.

- Обзор возможностей системы безопасности;
- Введение в объекты;
- Ограничение доступа;
- Настройка учетных записей пользователей Windows

5.1 Обзор возможностей системы безопасности

Принципы обеспечения безопасности системы Navision разработаны с целью предоставить вам полный контроль над тем, какие пользователи имеют право доступа к различным компонентам приложения. Функции безопасности могут быть разделены на три уровня: доступ к базе данных, доступ к объектам и доступ к записям.

1 На уровне доступа к базе данных вы можете контролировать, каким пользователям предоставлено право доступа к файлу базы данных с клиентской системы или с сервера. На этом уровне невозможно контролировать доступ к различным данным внутри системы. Для этого предназначены функции уровня доступа к объектам.

2 На уровне доступа к объектам вы можете контролировать доступ пользователей к различным объектам, включая права на работу с таблицами, например, для чтения, обновления, вставки и удаления.

3 На уровне доступа к записям администратор может предоставить пользователю доступ лишь к некоторым строкам таблицы, запрещая, таким образом, доступ к остальным строкам. Мы не будем здесь обсуждать этот уровень безопасности, так как для этого требуется глубокое понимание того, как работает приложение. Эта тема изложена в учебном курсе *Navision Essentials*.

Для настройки системы безопасности откройте меню Tools и выберите пункт Security. Меню Security содержат следующие опции.

Database logins : позволяет создавать учетные записи пользователей для регистрации при работе с базой данных. Эта настройка относится к уровню доступа к базе данных. Пользователь, имеющий такую учетную запись, может принадлежать к одной или нескольким ролевым группам (roles).

Windows logins : позволяет указывать пользователей Windows или группы пользователей Windows, которым также разрешена работа с базой данных. Эта настройка тоже относится к уровню доступа к базе данных. Пользователь, имеющий учетную запись Windows, может принадлежать к одной или нескольким ролевым группам.

Roles : позволяет создавать новые ролевые группы. Ролевая группа — это набор прав доступа, ассоциируемый обычно с

определенной работой или задачей. Существуют созданные по умолчанию типичные ролевые группы, которых достаточно для выполнения большинства возникающих задач. Перечень этих групп содержится в руководстве *Installation and System Setup*. Правом доступа называется разрешение получать доступ к объекту, а также уровень этого доступа. Когда вы распределяете пользователей по ролевым группам, вы определяете права доступа на уровне объектов.

Password : позволяет вам изменить пароль учетной записи, посредством которой вы зарегистрировались. Эта функция доступна лишь для учетных записей базы данных. Если вы зарегистрированы посредством учетной записи Windows, используйте средства Windows для изменения пароля.

5.2 Введение в объекты

Для понимания того, как действуют ограничения прав доступа в Navision, необходимо понимать предназначение объектов, к которым пользователям предоставляется доступ. В системе Navision существует всего семь типов объектов для настройки прав доступа. Следующая таблица содержит краткое описание каждого типа.

Тип объекта	Описание
Table Data (табличные данные)	Каждый раздел базы данных, относящийся к определенной компании, содержит собственные объекты типа «табличные данные». Для того чтобы пользователь мог просмотреть или обновить данные в таблицах, он прежде всего должен иметь доступ к соответствующему объекту типа « табличные данные».
Table (таблица)	Это описание таблицы. Оно определяет структуру и содержит код объекта типа «таблица». Объекты этого типа используют объекты типа «табличные данные» для манипуляций с данными.
Form (форма)	Объекты этого типа позволяют пользователю взаимодействовать с объектами типа «таблица». Пользователь не может работать непосредственно с данными в базе данных, он должен использовать форму. Форма — это графический интерфейс, дающий доступ к объекту типа «таблица» и с его помощью к табличным данным.
Report (отчет)	Предназначение объектов этого типа — предоставлять пользователю сводку данных или преобразовывать их для него. Выводимую информацию можно распечатать или просмотреть на экране.
Dataport (порт данных)	Объекты данного типа предназначены для экспортирования данных из таблиц или импортирования в них.
Codeunit (программный модуль)	Объекты этого типа позволяют пользователю запускать код, написанный на языке Client Application Language (C/AL). Таблицы, формы, отчеты и порты данных также включают код C/AL, однако объекты типа «программный модуль» предназначены только

	для этого.
System (система)	Этот объект дает вам возможность ограничивать доступ к различным пунктам меню приложения, а также к средствам, которые могут использоваться из программы.

Ограничивая доступ к различным объектам, вы можете настроить систему безопасности приложения в соответствии со своими специфическими требованиями. Как это сделать, обсуждается в последующих разделах.

5.3 Ограничение доступа

В этом разделе мы расскажем о том, как ограничить доступ пользователей к базе данных или к объектам, которые в ней содержатся.

Ограничение доступа к базе данных

В этом разделе мы создадим две учетные записи пользователей, зададим их пароли и установим для каждой учетной записи срок действия, по истечении которого доступ к базе данных для этих пользователей будет закрыт.

1 Запустите клиентское приложение Navision.

2 В меню **Сервис (Tools)** выберите пункт **Защита (Security)**, затем **Логины БД (Database Logins)**. Появится окно **Логины БД (Database Logins)**.

3 Выберите пустую строку и в полях **User ID** и **Name** введите *Super*, оставив остальные поля пустыми, в том числе и поле пароля.

4 Выберите следующую пустую строку, в полях **User ID** и **Name** введите *Student* , в поле **Password** введите *password* , поле **Expiration Date** оставьте пустым.

Примечание

Используйте имена пользователей и пароли, указанные в тексте. Если вы не будете следовать этому правилу и в дальнейшем забудете заданные пароли, вам придется использовать другую базу данных.

Термин «пользователь» (user) используется в тексте для обозначения пользователя системы Navision. Помните, что пользователь может иметь собственную учетную запись для доступа к базе данных, либо право на доступ к базе данных может быть предоставлено на основе данных его учетной записи Windows.

После того как вы создали учетные записи пользователей базы данных, доступ к базе данных будет предоставляться только этим двум пользователям. Однако вы все еще зарегистрированы в базе данных как пользователь с системными привилегиями (super user) без имени.

5 Закройте приложение Navision и запустите его вновь. Появится приглашение ввести имя и пароль пользователя.

6 В поле **User ID** введите *Super* , поле **Password** оставьте пустым. Щелкните OK.

Посмотрите на строку состояния. В ее правом нижнем углу показано имя пользователя, под которым вы зарегистрированы — *Super*. Пока оба пользователя, и *Super*, и *Student* имеют одинаковые права доступа к базе данных — неограниченный доступ. Каждый из них может удалить учетную запись другого, по сути, лишая того доступа к базе данных.

Ограничение доступа к объектам

Теперь необходимо ограничить доступ пользователя *Student* объектами, имеющими отношение к информации о клиентах. Этому пользователю разрешается лишь добавлять, удалять и обновлять записи о клиентах. Речь идет об объекте типа «табличные данные», содержащем таблицу **Customer**. Кроме того, чтобы иметь возможность просматривать табличные данные, пользователю необходим доступ к формам и таблицам.

Если пользователь системы Navision является членом группы, имеющей право доступа к объекту, то он также имеет право доступа к этому объекту, даже если пользователь одновременно принадлежит к другой группе, у которой такого права нет.

Сейчас вам предстоит распределить пользователей *Super* и *Student* по ролевым группам. Первая ролевая группа, в которую следует добавить пользователей, называется Super. Ее членам предоставляется полный доступ ко всем объектам. В системе Navision должен существовать по меньшей мере один пользователь с системными привилегиями доступа к базе данных (super user). До того как пользователи распределены между ролевыми группами, все они обладают системными привилегиями.

1 В меню **Сервис (Tools)** выберите пункт **Защита (Security)**, затем **Логины БД (Database Logins)**.

2 Выберите строку для пользователя *Super* и щелкните **Роли (**Roles).

Появится окно **Роли (Roles)**, содержащее перечень всех ролевых групп, к которым принадлежит данный пользователь. Сейчас *Super* не принадлежит ни к одной из групп.

3 Выберите первое поле в пустой строке и нажмите кнопку **AssistButton**, чтобы выбрать ролевую группу. Появится список, содержащий все созданные по умолчанию ролевые группы. Выберите группу Super и щелкните OK.

Пользователь *Super* является отныне пользователем с системными привилегиями. Нет необходимости приписывать его к каким-либо другим ролевым группам. Принадлежность к группе Super дает ему права полного доступа к системе.

4 Закройте окно **Роли (Roles)** и выберите пользователя *Student*. Щелкните **Роли (Roles)**, чтобы выбрать для него ролевые группы.

5 Выберите первое поле в пустой строке и нажмите кнопку **AssistButton**, чтобы выбрать ролевую группу. Появится список, содержащий все созданные по умолчанию ролевые группы. Выберите группу All и щелкните OK. К этой ролевой группе должны принадлежать все пользователи, не входящие в группу Super. Ролевая группа All имеет права доступа к большей части объектов. Необходимо лишь дополнительно предоставить доступ к табличным данным и, возможно, к системным функциям.

Теперь пользователь *Student* имеет ограниченный доступ к системе. Эти ограничения достаточно сильны, поэтому *Student* может работать лишь с весьма небольшим набором данных. Давайте убедимся в этом.

6 Закройте клиентскую программу и запустите ее вновь. В поле **User ID** приглашения введите *Student*, в поле **Password** введите пароль, затем щелкните OK. Если при регистрации возникли проблемы, зарегистрируйтесь как *Super* и заново задайте пароль для пользователя *Student*.

7 Если не появилось главное меню, нажмите клавишу F12.

8 Попытайтесь из главного меню вызвать какие-нибудь другие окна. Щелкните **Sales and Receivables**, затем **Customers**. Вы должны получить сообщение об ошибке, говорящее об отсутствии прав доступа.

9 В меню **Sales and Receivables** щелкните **Setup**, затем **Sales & Receivables Setup**. Вы можете просмотреть данные в этой форме. Однако если вы попытаетесь отредактировать что-либо и закрыть форму, вы обнаружите, что данный пользователь не обладает правами на внесение изменений даже в эти данные.

Теперь необходимо предоставить пользователю *Student* права создавать, удалять и модифицировать записи клиентов. Для этого можно было бы воспользоваться стандартными ролевыми группами, но для выполнения данного упражнения мы создадим новую группу.

1 Закройте клиентскую программу и запустите ее вновь. В поле **User ID** приглашения введите *Super*, поле **Password** оставьте пустым, щелкните OK

2 В меню **Сервис (Tools)** выберите пункт **Защита (Security)**, затем **Роли (Roles)**. В окне **Роли (Roles)** приведен перечень всех существующих ролевых групп.

3 Нажмите клавишу F3, чтобы вставить пустую строку.

4 В поле **Role ID** введите *CUSTOMER-IMD*, в поле **Name** введите *Customer Table – IMD*. Чтобы сохранить новую ролевую группу, необходимо перевести курсор в любую другую строку. Нажмите клавишу со стрелкой ВВЕРХ, затем ВНИЗ. Курсор должен располагаться в строке, которую вы только что добавили. После того, как новая ролевая группа была сохранена, можно продолжить.

5 Следующий шаг — определить права доступа для этой группы. В окне **Roles** щелкните Роль (Role), затем Права Доступа (Permissions). Появится окно **Права Доступа (Permissions)**.

6 В первой пустой строке введите следующие значения.

Object Type	Table Data
Object ID	18
Object Name	(это поле заполняется автоматически)
Read Permission	Yes
Insert Permission	Yes
Modify Permission	Yes
Delete Permission	Yes
Execute Permission	(Изменить значение в этом поле нельзя)

7 Закройте окно **Permissions**. Убедитесь в том, что в окне **Roles** выбрана ролевая группа, щелкните **Role**, затем **Database Logins**. В этом окне показаны все учетные записи пользователей базы данных, ассоциированных с этой группой. Конечно, поскольку это новая ролевая группа, список учетных записей, отнесенных к этой группе, пуст.

8 В поле **User ID** выберите или введите имя пользователя *Student*. Это еще один способ добавить учетную запись пользователя к ролевой группе. Прежде мы добавляли ролевые группы к учетным записям пользователей.

9 Закройте окно **Логины БД (Database Logins)**, затем закройте окно **Роли (Roles)**.

Теперь *Student* должен иметь доступ к данным таблицы **Customer**. Проверим это, зарегистрировавшись в системе под именем *Student*.

1 Закройте клиентскую программу и запустите ее вновь. В поле **User ID** приглашения введите *Student*, в поле **Password** введите пароль, затем щелкните OK. Если при регистрации возникли проблемы, зарегистрируйтесь как *Super* и заново задайте пароль для пользователя *Student.*

Если не появилось главное меню, нажмите клавишу F12.

2 Попытайтесь из главного меню вызвать какие-нибудь другие окна. Щелкните **Sales and Receivables**, затем **Orders**. Вы должны получить сообщение об ошибке, говорящее об отсутствии прав доступа. Однако доступ к **Customers** должен быть открыт.

3 Щелкните **Customers**. Должна появиться карточка **Customer**. Теперь у вас должна быть возможность добавлять (F3), модифицировать или удалять записи (не удаляйте никаких записей из базы данных).

Теперь вы знаете способы ограничения прав доступа пользователей как на уровне базы данных, так и на уровне содержащихся в ней объектов.

Ограничение доступа к записям

При использовании SQL Server система Navision позволяет производить еще более тонкую настройку существующих ролевых групп посредством применения фильтров безопасности. Это позволяет максимально адаптировать систему безопасности для нужд вашей организации. Фильтры безопасности можно использовать, чтобы препятствовать доступу неавторизованных пользователей к конфиденциальной информации. Вы можете, например, ограничить доступ пользователя к определенным счетам, учетным периодам, аналитическим измерениям или к информации, касающейся определенных клиентов или поставщиков.

Если вы применяете фильтр безопасности к ролевой группе, то при этом вы изменяете эту группу. Следовательно, все пользователи, которые будут отнесены к данной группе, будут обладать правами доступа с учетом изменений. Если вы не желаете, чтобы измененные права доступа применялись ко всем остальным пользователям, которые будут присоединены к данной ролевой группе в дальнейшем, необходимо рассмотреть возможность создания ролевой группы с новым уровнем безопасности перед тем, как вносить изменения в стандартный уровень безопасности ролевой группы. Кроме того, может потребоваться создание новой ролевой группы, которая содержит фильтры безопасности, соответствующие требованиям вашей компании. Каждое подразделение может обладать собственным рядом ролевых групп, каждая из которых имеет индивидуальный набор фильтров безопасности. Если кто-либо входит в систему, когда вы вносите изменения для данной ролевой группы, то он не будет проинформирован о произведенных изменениях до тех пор, пока не выйдет и вновь не войдет в систему.

Установка одного фильтра безопасности сама по себе не означает, что пользователь может видеть только те записи, которые определены данным фильтром. Каждый пользователь, как правило, будет относиться более чем к одной ролевой группе в текущей базе данных, и будет обладать правами доступа, определенными в каждой из этих ролевых групп. Следовательно, права доступа, которыми владеет пользователь, являются суммой всех прав доступа, определенных для каждой ролевой группы, к которой пользователь будет присоединен.

Если пользователь относится к двум ролевым группам, которые предоставляют права доступа к одной и той же таблице, и обе ролевые группы имеют фильтры безопасности, наложенные на эту таблицу, то в таком случае применяется сумма фильтров безопасности. Это означает, что если одним фильтром определено право пользователя лишь на чтение записей с 1 по 10, а другим фильтром установлено право вносить изменения в записи с 5 по 20, то пользователь сможет изменять записи с 5 по 20 и будет иметь право только на чтение записей с 1 по 20.

Важно помнить, что отсутствие фильтра безопасности у какой-либо таблицы означает, что пользователь может, например, прочитать все записи в данной таблице. Это значит, что если права доступа в отношении определенной таблицы предоставляются пользователю через более, чем одну ролевую группу, то фильтр безопасности для этой таблицы, наложенный в одной ролевой группе, не будет иметь силы, если другая ролевая группа предоставляет пользователю разрешение производить те же самые операции в той же самой таблице, но без каких-либо фильтров.

Чтобы применить фильтр безопасности, выполните следующие действия.

1 Нажмите **Сервис(Tools)**, затем **Защита(Security)**, затем **Роли(Roles)** и выберите ролевую группу, к которой вы хотите применить фильтр.

2 Щелкните **Права Доступа(Permissions)**, появится окно **Права Доступа(Permissions)** для выбранной ролевой группы.

3 В поле **Тип объекта (Object Type)** нажмите кнопку **AssistButton** и выберите **Table Data**.

4 В поле **Объект ID(Object ID)** нажмите кнопку **AssistButton** и выберите таблицу, к которой вы хотите применить фильтр.

5 Присвойте ролевой группе права доступа **Чтение(Read)**, **Вставка(Insert)**, **Изменение(Modify)**, **Удаление(Delete)** и **Выполнение(Execute)**, которые вы бы хотели применить к данной ролевой группе.

6 В поле **Доступ фильтр (Security Filter)** нажмите кнопку **AssistButton**. Появится стандартное окно **Фильтр таблицы(Table Filter)**.

7 Для создания фильтра, воспользуйтесь **AssistButton** в колонке **Поле(Field)**,.чтобы выбрать поле, которое нужно использовать для данного фильтра.

8 В поле **Фильтр(Filter)**, введите значение, которое вы хотите использовать в данном фильтре.

9 Нажмите OK, чтобы применить фильтр.

Повторите эту процедуру для всех таблиц, к которым нужно применить фильтр. Если нужно использовать тот же критерий вновь, вы можете вырезать и вставить его или использовать клавишу F8.

5.4 ОПРЕДЕЛЕНИЕ ПРАВ ДОСТУПА

Давайте изучим процесс предоставления прав доступа ролевой группе более подробно. Вначале, рассмотрим существующую ролевую группу.

1 Закройте клиентское приложение и вновь откройте его. После этого, введите *Super* в поле **User ID**, оставьте поле **Password** пустым и нажмите OK.

2 Нажмите **Сервис(Tools)**, затем **Защита(Security)**, затем **Роли(Roles)**. Вы увидите список всех ролевых групп, определенных в системе.

3 Выберите ALL (все ролевые группы), нажмите кнопку **Роль(Role)**, а затем **Права Доступа(Permissions)**.

Вы увидите список прав доступа, которые были присвоены данной ролевой группе. Опустите курсор к концу списка.

Посмотрите на объекты в конце списка. Это объекты типа System. Каждый объект System, который вы видите, корреспондирует с пунктом меню **Main Menu**. Обратите внимание, что ВСЕ пользователи имеют доступ только к нескольким таким объектам. Существует ряд дополнительных функций, таких как **File, Company, Open**, к которым имеет доступ каждый пользователь, независимо от принадлежности к ролевой группе.

Выберите поле **Объект ID(Object ID)** в любой строке, которая имеет в качестве типа объекта (Object Type) -- System. Щелкните по кнопке **Lookup**, которая появится в этом поле, и изучите различные объекты. Обратите внимание, что большинство из них, как уже упоминалось, корреспондирует с пунктами в **Main Menu**. Некоторые из них, тем не менее, соотнесены с другими функциями, например выборочным доступом к развитию системы, и разрешением пользователю получения доступа к Navision через C/ODBC. Закройте эту форму, нажав кнопку Cancel.

Далее, поднимите курсор немного вверх в окне **Права доступа (Permissions)** ко ВСЕМ пользователям. Обратите внимание, что пять строк, дают права доступа к пяти различным типам объектов с **Объект ID (Object ID)**, установленным на 0. Это значит, что указанные права доступа были присвоены всем объектам данного типа. Например, ВСЕ пользователи имеют возможность работать с любыми объектами типа Report. Фактически, у всех пользователей есть возможность работать

со всеми объектами Table. Тем не менее, запомните, что способность работать с объектами Table, не дает права доступа к данным, так как они являются другим объектом -- Table Data.

Смысл этого заключается в том, что если вы хотите предоставить пользователю право доступа к конкретной форме и запретить право доступа к использованию этой формы для других пользователей, вы должны сначала удалить все строки, которые предоставляют ВСЕМ пользователям права доступа ко всем формам. Затем вы должны добавить использование прав доступа для всех иных форм, таким образом, чтобы ВСЕ пользователи могли иметь доступ к пунктам меню и прочим формам, не имеющим защиты. Кнопка **Все Объекты(All Objects)** может помочь в этом. В заключение, вы можете создать ролевую группу, содержащую форму, которую вы хотите защитить, и связать эту ролевую группу только с пользователями, которым вы хотите предоставить возможность работы с этой формой.

Далее, опустите немного курсор в окне **Права Доступа(Permissions)** так, чтобы вы могли видеть права доступа в Table Data, которые применяются ко ВСЕМ пользователям. В некоторых из них, например, таблице данных **Sales & Receivables Setup**, пользователю разрешено лишь чтение (Read) данных, но не обновление их каким-либо способом. В других, например, таблице данных **Report List**, пользователю разрешено чтение, изменение, вставка и удаление данных.

Посмотрите на права доступа для объекта **No. Series Table Data**. Обратите внимание, что эта ролевая группа имеет право доступа на чтение (Read), установленное в **Да(Yes)**, так что ВСЕ пользователи могут читать данную таблицу. Однако отметьте, что право доступа на изменение (Modify) установлено в Indirect. Это означает, что хотя пользователь не может войти в эту форму и изменить данные непосредственно, он может сделать это через другой объект, к которому у него есть доступ на изменение, и внести изменения косвенным путем. Без косвенного права доступа, ВСЕ пользователи не имеют возможности создавать следующий номер в числовой последовательности и поэтому не могут, например, создавать следующий номер **Sales Order**.

Теперь вы можете рассмотреть некоторые другие ролевые группы, чтобы понять, как задаются права доступа, и затем поупражняться с вашей ролевой группой *CUSTOMER-IMD*, чтобы понять, как добавление различных прав доступа дает возможность для пользователя *Student* производить различные действия.

Примечание

Обратите внимание, что в Navision не предполагается определение уровня безопасности через систему безопасности. Определение уровня безопасности обычно происходит с помощью построения форм.

Ограничение ролевых групп в рамках компании

Необходимо обсудить еще один вопрос, касающийся ролевых групп и пользователей. Это возможность предоставлять пользователям ограниченные права в рамках одной компании.

1 Закройте клиентскую программу и вновь откройте ее. После этого, введите *Super* в поле **User ID,** поле **Password** оставьте пустым и нажмите OK.

2 Нажмите **Сервис(Tools)**, **Защита(Security)**, **Логины БД(Database Logins).**

3 Выберите пользователя *Student* и нажмите **Роли(Roles)**.

   Отметьте, что пользователь *Student* принадлежит к ролевой группе CUSTOMER-IMD. Это дает возможность вставлять, изменять и удалять записи о клиентах. В данный момент, ограничения для конкретной компании отсутствуют (поле **Company** является пустым).

4 В поле **Company**, нажмите кнопку **Lookup**. Теперь мы можем видеть список компаний, находящихся в базе данных.

   В данный момент у нас есть только одна компания. Давайте ограничим работу пользователя *Student* этой компанией.

5 Щелкните дважды по компании CRONUS или выберите ее и нажмите OK.

Теперь пользователь *Student* ограничен возможностями чтения, вставки, изменения или удаления записей о клиентах компании CRONUS International Ltd. Если мы создаем новую компанию, пользователь *Student* не сможет читать, вставлять, изменять или удалять записи о клиентах в новой компании.

5.5 Изменение пароля

Бывают случаи, когда пользователь хочет изменить свой пароль. Хотя вы не можете вынудить пользователя изменить пароль в системе Navision, вероятно вам потребуется, чтобы пользователи изменяли пароли на регулярной основе. Для того чтобы изменить пароль в Navision, необходимо выполнить следующие действия.

1. Нажмите **Сервис(Tools)**, затем **Защита(Security)**, затем **Пароль(Password)**.
2. Введите текущий пароль (тот же самый пароль, который вы используете для открытия базы данных).
3. Введите новый пароль в полях **Новый Пароль (New Password)** и **Повторите новый пароль(Reenter New Password)**. После ввода этого значения дважды вы будете гарантированы от ошибочного ввода. Это необходимо по причине того, что вы не видите пароли, которые вводите.
4. Нажмите OK. До тех пор, пока новые пароли, которые вы вводите, не совпадут, программа не изменит ваш пароль.

Эта процедура позволяет любому пользователю изменить свой пароль без помощи администратора.

5.6 НАСТРОЙКА УЧЕТНЫХ ЗАПИСЕЙ ПОЛЬЗОВАТЕЛЕЙ WINDOWS

Чтобы настроить учетные записи Windows, необходимо быть подсоединенным к базе данных через сервер. По этой причине в данном курсе не рассматривалась эта задача. В операционной системе, установленной на вашем компьютере, это может осуществляться иным путем. Для получения наилучших результатов, ваша сеть должна использовать ОС Windows 2000, и если вы занимаетесь настройкой безопасности системы, на вашем компьютере также должна использоваться Windows 2000.

В диалоговом окне учетных записей Windows, нужно выбрать пользователей сети или группы, которым вы хотите предоставить доступ к базе данных Navision. При этом пароль или срок действия отсутствует, так как пользователи будут идентифицироваться и получать доступ через сетевой logon. Фактически пользователь не должен вводить User ID и пароль при входе в Navision. Кроме этого базового отличия, учетные записи Windows действуют так же, как учетные записи базы данных, и все примеры, приведенные выше, могут быть повторены для учетных записей Windows.

Примечание

Даже если все пользователи будут входить в Navision с помощью учетной записи Windows, вы ДОЛЖНЫ иметь по крайней мере одну настройку для пользователя в диалоговом окне учетных записей базы данных. Этот пользователь необходим, чтобы сохранять безопасность базы данных и иметь хотя бы один способ войти в базу данных локально (не через сервер)

Для получения дополнительной информации о безопасности, обращайтесь к руководству *Installation and System Management.*

5.7 НЕСКОЛЬКО СЛОВ О ЛИЦЕНЗИРОВАНИИ

Важно отметить, что пользователи ограничены не только ролевыми группами, членами которых они являются, но также и лицензионным файлом, который они используют. Например, если клиент не покупает гранулу Inventory, предоставляющую пользователям право доступа к вставке, изменению или удалению записей в таблицу товаров, он не будет иметь возможности использовать эту таблицу

Диагностика ошибки, связанной с правами доступа

Чтобы провести диагностику ошибки, связанной с правами доступа, прежде всего нужно убедиться, что причиной ошибки не является лицензионный файл. Следующие процедуры помогут в этом.

Ситуация: пользователь пытается произвести действие и получает сообщение об ошибке, что он не имеет права доступа к таблице.

1 Войти как пользователь superuser и попытаться произвести то же действие заново. Если результат положительный, то проблема не связана с лицензионным файлом. Если результат отрицательный, выполните шаг 2.

2 Нажмите **Сервис(Tools)**, затем **Лицензия(License Information)**. Появится окно **Информация о лицензиии (License Information)**.

3 Нажмите **Изменить(Change)**. Это позволит изменить лицензию разработчика (solutions developer's license). Найдите ваш лицензионный файл и нажмите **Open**.

4 Войдите как пользователь superuser еще раз и попробуйте выполнить то же самое действие. Если результат положительный, выполните шаг 5. Если результат отрицательный, вам возможно потребуется обратиться в службу поддержки.

5 Войдите как пользователь, у которого возникла вышеуказанная проблема и попробуйте выполнить то же самое действие. Если результат положительный, то проблема связана с лицензионным файлом. Если результат отрицательный, проблема в другом.

Если после вышеупомянутых шагов вы определили, что проблема заключается в лицензионном файле, вы должны определить, какие гранулы необходимо приобрести, чтобы иметь возможность осуществить нужное действие. Если вы определили, что проблема не

связана с лицензионным файлом, необходимо рассмотреть настройки безопасности и убедиться, что пользователь принадлежит к соответствующим ролевым группам, и что эти группы имеют надлежащие права доступа. Помните, что данная система безопасности использует наименее эффективный метод ограничений, чтобы определять права доступа.

Глава 6

Функциональные модули и гранулы

Данная глава содержит в себе обзорную информацию о принципах совместной работы компонентов, входящих в состав приложения Navision. Информация, приведенная в данной главе, позволит вам быстро выделять наиболее важные части функциональных модулей приложения. Также вы сможете создавать свои собственные функциональные модули, выглядящие и работающие подобно стандартным.

Эта информация будет особенно полезна вам при разработке с помощью Navision приложений, взаимодействующих с уже существующими стандартными функциональными модулями, а также при изменении уже существующих приложений. Создание и использование несложного проекта существенно снизит вероятность серьезной ошибки при внесении изменений.

Эта глава состоит из следующих разделов:

- Функциональные модули Navision;
- Основные компоненты функционального модуля;
- Стандартные зависимости в функциональном модуле;
- Перемещение по функциональному модулю.

6.1 Функциональные модули Navision

Функциональный модуль (functional area) представляет собой набор бизнес-функций, объединенных в рамках Navision. В состав Navision входят такие функциональные модули, как General Ledger (Финансы) , Sales & Receivables (Продажи и клиенты) и Inventory (Товары). Каждый функциональный модуль состоит из одной или нескольких гранул.

Основными составляющими каждого функционального блока являются журнал, с помощью которого осуществляются все транзакции, процедура записи данных и учетная книга, содержащая в себе все записанные данные. Все отчеты генерируются на основе данных, содержащихся в учетной книге.

Внимание

При изучении дизайна функционального модуля следует принимать во внимание, что не только система Navision создана на основе этой модели, но и сама среда разработки C/SIDE включает в себя поддержку этой модели. Важно понимать, что это означает не только то, что вы можете спокойно вносить изменения в существующие функциональные блоки, но и то, что создаваемые вами функциональные блоки будут взаимодействовать с системой наиболее оптимальным образом.

Функциональные модули

Стандартное приложение включает в себя следующие функциональные модули:

1 General Ledger (Финансы).

2 Fixed Assets (Основные средства).

3 Sales & Receivables (Продажи и клиенты).

4 Relationship Management (Управление отношениями).

5 Service Management (Сервисный центр).

6 Purchases & Payables (Покупки и поставщики).

7 Inventory (Товары).

8 Warehouse Management* (Управление складом *).

9 Manufacturing (Производство).

10 Capacity Requirements Planning (Планирование производственных

мощностей).

11 Resources (Ресурсы).

12 Jobs (Работы).

13 Human Resources* (Управление кадрами *).

14 Commerce Portal.

*Эти модули включают в себя многие компоненты полноценного функционального модуля, но не все из них.

Гранулы

Функциональные модули состоят из более мелких элементов, называемых гранулами (granules). Гранула включает в себя часть функциональных возможностей функционального модуля и может быть приобретена отдельно. Другими словами, гранула содержит в себе набор функций, который вы можете либо приобрести, если испытываете необходимость в этих функциях, либо не приобретать и тем самым снизить общую стоимость системы. Каждый функциональный модуль состоит, по крайней мере, из одной гранулы (обычно называемой гранулой базового приложения этого модуля).

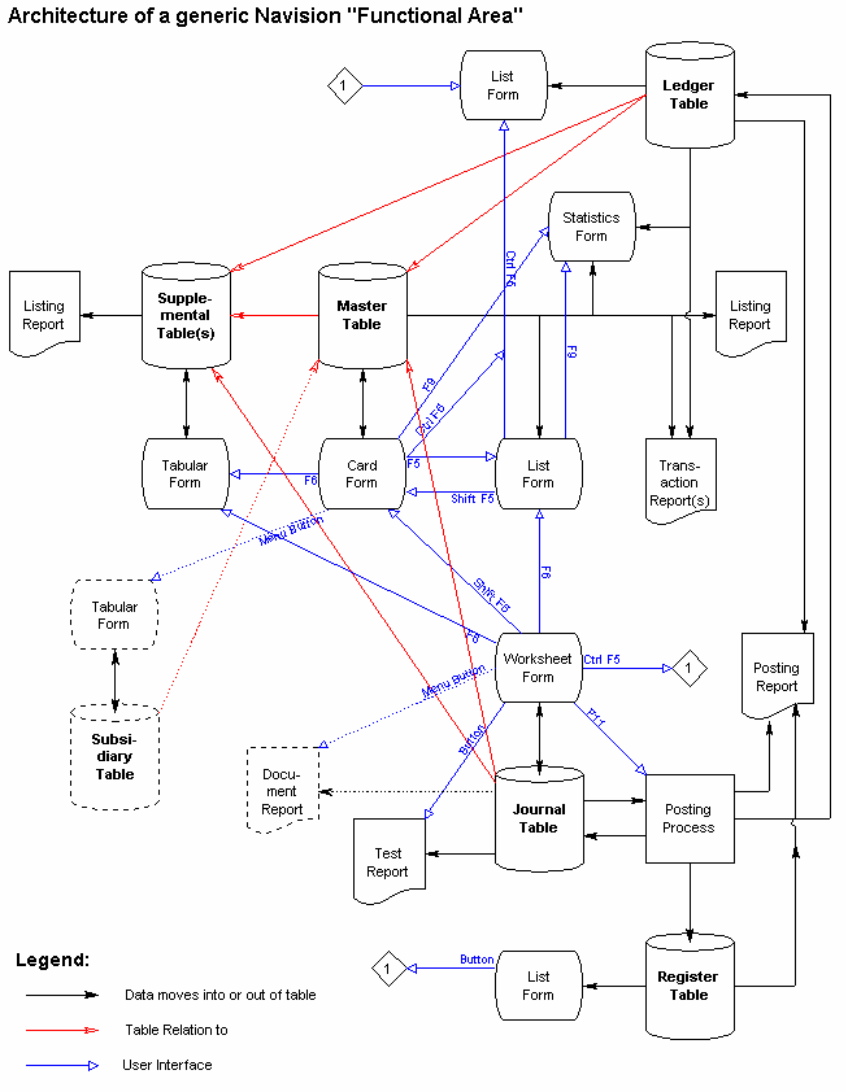
В состав базового приложения входит достаточно много гранул. Некоторые из гранул зависят от других. Например, вам придется приобрести гранулу Basic Inventory (Товары), если вы хотите использовать гранулу Bills of materials (Комплекты). Более подробная информация о гранулах и их зависимостях содержится в классах приложения и продаж.

Дополнения

Дополнения (add-ons) представляют собой гранулы, разрабатываемые вне базового приложения. Дополнения могут содержать в себе функции для конкретного вертикального рынка, либо расширять функциональные возможности уже существующих в Navision функциональных модулей. Дополнения обычно разрабатываются отдельными Центрами решений Navision.

6.2 Основные компоненты функционального модуля

Функциональный модуль состоит из нескольких различных типов связанных объектов. Каждый тип объектов обладает определенными параметрами и служит для выполнения определенных функций в рамках функционального модуля. Ниже приводится краткое описание этих компонентов.



Основные таблицы

Каждый функциональный модуль обладает одной, реже двумя основными таблицами (master tables). Каждая запись в основной таблице описывает объект, лицо или товар, в отношении которого совершаются транзакции.

Например, основной таблицей функционального модуля Inventory является таблица **Item** (**Товары**), поскольку все осуществляемые транзакции совершаются по отношению к товарам, уменьшая или увеличивая имеющееся в наличии количество этих товаров.

1 Щелкните **Товары(Inventory)**, **Товары(Item)** и на экране появится окно **Карточка товара (Item Card)**.

Карточка товара (Item Card) содержит разнообразную информацию о каждом товаре. Теперь давайте перейдем к рассмотрению другого компонента этого функционального модуля.

2 Закройте окно **Карточка товара (Item Card),** щелкните **Журналы Товаров (Item Journals)** и на экране появится окно **Журнал Товаров (Item Journal)**.

С помощью этого окна вы можете осуществлять транзакции, относящиеся к товарам. Обратите внимание на поле **Товар Но. (Item No.).** Это поле необходимо для осуществления записи любой из этих транзакций, поскольку каждая транзакция в модуле Inventory должна затрагивать, по крайней мере, один товар.

Учетные таблицы

Каждый функциональный модуль обладает одной, реже двумя учетными таблицами (ledger tables). Каждая запись в учетной таблице представляет собой записанную транзакцию или ее часть.

Другими словами, транзакция может состоять из одной или нескольких записей.

1 Закройте окно **Журнал Товаров (Item Journal)** и щелкните **Товары (Items)**.

2 Щелкните **Карточка товара (Item Card)** и нажмите клавишу F5, чтобы просмотреть список товаров. Найдите товар 70000 и щелкните OK.

3 Нажмите CTRL+F5 чтобы просмотреть записи учетной таблицы, относящиеся к этому товару. Существует несколько таких записей.

4 Попробуйте щелкнуть кнопку **Navigate**, выделив мышью какую-нибудь из этих записей, чтобы узнать, какие транзакции были осуществлены совместно с транзакцией, описываемой данной записью.

Каждая запись в учетной книге содержит информацию только о той части транзакции, которая затрагивает данный функциональный модуль. Пользователь не может добавлять записи непосредственно в учетную таблицу, поскольку она является “защищенной”.

Примечание

Защищенная таблица -- это таблица, прямое внесение изменений в которую возможно только в том случае, если вы обладаете лицензией Navision Solution Developer. В противном случае содержимое таблицы может изменяться только другими объектами, обладающими правами на внесение изменений в эту таблицу.

Журнальные таблицы

Каждый функциональный модуль обладает одной или несколькими журнальными таблицами (journal tables). Журнальные таблицы служат для ввода транзакций пользователем.

1 Щелкните **Товары (Inventory)** в **Main Menu (Главном меню)**.

2 Щелкните **Журнал Товаров (Item Journals)** и на экране отобразится окно **Журнал Товаров (Item Journal)**.

С окном **Журнал Товаров (Item Journal)** мы уже встречались ранее. С помощью этого окна создаются записи в учетной книге. До тех пор пока эти транзакции находятся в журнале, пользователь может изменять их, удалять, добавлять снова и т.д. Однако после осуществления записи транзакций (т.е. после внесения изменений в учетную книгу), транзакции становятся постоянными и более не могут быть изменены.

Первичный ключ таблицы **Journal (Журнал)** состоит из трех частей, **Journal Template Name (Код Шаблона Журнала), Journal Batch Name (Код Раздела Журнала)** и **Line No. (Строка Но.).** Первые два поля служат для связи с дополнительными таблицами, содержащими в себе соответственно список шаблонов и список пакетов журнальных операций (journal batches). Значение, содержащееся в поле **Строка Но. (Line No)**, представляет собой целое число, определяющее конкретную строку журнала. Такая структура первичного ключа этой таблицы позволяет вам создавать большое количество журналов и пакетов журнальных операций.

Значительная доля алгоритмов реализации деловых операций

функционального модуля сосредоточена в триггерном коде журнальной таблицы. При вводе данных пользователем осуществляется проверка этих данных и перекрестная проверка с уже введенными данными. На основании итогов этих проверок вычисляются некоторые результаты.

Упражнение 1 **Ввод транзакций**

Попробуйте ввести несколько транзакций. Изменяйте число товаров как в большую, так и в меньшую сторону. Проверьте, можете ли вы записать эти транзакции.

Если вам удалось записать несколько транзакций, попробуйте найти записи в учетной таблице, относящиеся к этим транзакциям. Для этого перейдите в окно **Карточка товара (Item Card)** данного товара хранения.

Программист, вносящий запись в журнал (например, из вновь созданного функционального модуля), может воспользоваться этими алгоритмами реализации деловых операций, вызвав для критических полей функцию “VALIDATE”.

Процедура записи данных

Каждый функциональный модуль использует свою собственную процедуру записи данных.

Процедура записи представляет собой функцию. Имя этого модуля обычно образуется добавлением “-Post Line ” к сокращенному названию журнала.

За незначительными исключениями процедура записи содержит в себе остальную часть алгоритмов реализации бизнес-правил.

Эта функция служит шлюзом между журналом и учетными таблицами данного функционального модуля. Данные поступают в учетную таблицу только через процедуру записи, а в процедуру записи данные попадают только через журнал.

Давайте найдем функцию, которую мы только что использовали для записи транзакций из журнала товаров.

1 Откройте **Object Designer (Дизайнер объектов)** и щелкните **Codeunits (Программные модули)**. На экране отобразится список функций, находящихся в базе данных.

2 Найдите **Codeunit 22 (Модуль 22)** (все функции упорядочены по номеру), **Item Jnl.-Post Line**. Именно эта функция создала записи в учетной таблице на основании записей в журнале.

Обратите внимание на **Codeunit 21** и **23** (**Модули 21** и **23**) - **Item Jnl.-Check Line** и **Item Jnl.-Post Batch**. Эти функции также участвуют в процессе записи данных.

Важная роль Общего журнала (General Journal)

Одним из ключевых моментов для понимания Navision является важная роль таблицы **General Journal Line**.

Эта таблица является журнальной таблицей сразу для нескольких функциональных модулей, в том числе для модулей:

1 General Ledger (Финансы).

2 Accounts Payable (Счета дебиторов).

3 Accounts Receivable (Счета кредиторов).

Несмотря на то, что каждый из этих модулей обладает своей собственной основной таблицей и своей учетной таблицей, все они используют один и тот же журнал и одну процедуру записи (**Gen. Jnl.-Post Line**, **Codeunit 12** (**Модуль 12**)).

Внесению изменений в учетную таблицу любого из этих модулей всегда предшествует автоматическое внесение изменений в **General Ledger (G/L)**.

Значение поля **Entry No. (Операция Но.)** таблицы **G/L Entry** обновляется при внесении изменений в любую другую учетную таблицу, поэтому данные в этих таблицах всегда согласованы и непротиворечивы.

Единственный во всей системе Navision анализ данных на непротиворечивость производится программным модулем 12. В ходе этого анализа проверяется, что каждый процесс записи, осуществляющийся с помощью модуля 12 (т.е. практически все

процессы записи), обладает сбалансированным сальдо пассивов и активов. Эта проверка позволяет обнаруживать ошибки в программном коде до того, как они могут привести к повреждению транзакционных данных, постоянно хранящихся в General Ledger.

Документы

Документы входят в состав не всех функциональных модулей. Но, так как все наиболее интенсивно используемые модули их имеют, о документах нельзя не упомянуть.

Документ представляет собой несложный для изучения пользовательский интерфейс. В общем случае, документ состоит из заголовка и нескольких строк записей. Обычно в наличии имеется форма, позволяющая одновременно изменять и заголовок, и записи. Вместе эти два элемента образуют документ. По существу, документ представляет собой удобный способ одновременного внесения большого количества записей в журналы нескольких различных функциональных модулей.

На самом деле документы представляют собой весьма сложные объекты. Документы объединяют в себе функциональные возможности основной и журнальной таблицы. Запись одного документа может привести к появлению десятков или даже сотен новых записей в учетных таблицах нескольких функциональных модулей.

Использование этого интерфейса Navision позволяет облегчить работу пользователей.

Аналитические измерения

В Navision 3.00 появился новый ключевой компонент архитектуры приложения, называемый аналитическими измерениями (dimensions). Измерения создаются на основе набора взаимосвязанных дополнительных таблиц. Это делается с помощью подменю **Setup (Настройка)** меню **General Ledger**. Каждое из измерений, связываемых с транзакциями на различных стадиях, присоединяется с помощью специальной вспомогательной таблицы. Эта таблица является вспомогательной для всех таблиц определенного типа.

Для всех журнальных таблиц такой вспомогательной таблицей является таблица **Journal Line Dimension.** Первичный ключ этой таблицы состоит из поля **Table ID (Идентификатор таблицы)**,

содержащего целочисленное значение, а также полей **Journal Template Name**, **Journal Batch Name** и **Line No.**, определяющих первичный ключ всех журнальных таблиц. Также существует поле **Allocation Line No.(Распределение Строка Но.)**, предназначенное для определения размещения в (размещения из) **General Ledger** и **Dimension Code** (коде аналитического измерения). Таким образом, для каждой журнальной записи, в любой таблице, шаблоне или пакете операций (batch) определен список **Dimension Codes**. Для каждого **Dimension Code** приведено выбранное значение **Dimension Value** (значение аналитического измерения).

Для всех учетных таблиц вспомогательной является таблица **Ledger Entry Dimension (Книга Операций Измерение)**. Первичный ключ этой таблицы состоит из поля **Table ID (Идентификатор таблицы)**, содержащего целочисленное значение, а также полей **Entry No.** и **Dimension Code (Код измерения)**. Таким образом, для каждой записи в любой из учетных таблиц определен список **Dimension Codes**. Для каждого **Dimension Code** приведено выбранное значение **Dimension Value** (значение аналитического измерения).

Аналогичным образом, для всех таблиц документов (заголовков и записей), вспомогательной является таблица **Document Dimension (Документ Измерение)**. Аналогичным образом, для основных таблиц вспомогательной является таблица **Default Dimension (Стандартное Измерение)**. Эта таблица используется для создания аналитических измерений по умолчанию для каждой записи в основной таблице.

Другие компоненты

В состав каждого функционального модуля входят другие таблицы и формы, в том числе различные дополнительные и вспомогательные таблицы. Они используются для группировки транзакционных данных, либо для вычисления различных величин, используемых при вводе или записи данных. Многие дополнительные таблицы, такие как **Currency (Валюта)** или **Payment Terms (Формула Оплаты)**, используются несколькими функциональными модулями.

Каждый функциональный модуль обладает своим собственным меню, вызвать которое можно из **Main Menu (Главное меню)**.

Первый столбец меню функционального модуля содержит:

1 Основную таблицу или таблицы.

2 Все важные дополнительные таблицы.

3 Журналы, документы и формы для ввода данных.

Второй столбец меню функционального модуля содержит:

1 Подменю печати отчетов и документов.

2 Формы регистрационных журналов.

3 Область перемещения.

4 Подменю **Periodic Activities (Периодические действия)**, предоставляющее доступ к регламентным журналам (Recurring journals), а также к дополнительным вычислительным процедурам и процедурам обновления данных, которые должны выполняться периодически. К числу таких процедур относится периодическое сжатие учетных таблиц и регистрационных журналов.

5 Внизу правого столбца находится подменю **Setup (Настройка)**, которое предоставляет вам доступ к настроечным и другим дополнительным таблицам.

Отчеты

Наконец, в Navision существуют объекты типа “отчет” (report).

Одним из наиболее важных отчетов для любого функционального модуля является отчет **Transaction Detail** (или **Account Detail**) (**Детализация транзакций**, или **Подробный отчет**). Этот отчет содержит в себе список записей всех основных таблиц и совершенных транзакций, отвечающих этим записям. Применяя соответствующие фильтры и ключи сортировки, пользователь может извлечь из этого отчета много разнообразной и полезной информации.

Другим очень важным отчетом является тестовый отчет для журнала. Пользователь должен иметь возможность сгенерировать тестовый отчет для каждого журнала, чтобы просмотреть результаты записи журнальных данных. При работе с большинством функциональных модулей вы можете сгенерировать тестовый отчет, щелкнув кнопку **Posting (Записать)** в форме журнала. Также этот отчет может быть использован для обнаружения ошибок, которые могут возникнуть, если вы попытаетесь записать пакет журнальных операций (journal batch).

Дополнительная информация

Вместе с этим курсом вы должны были получить несколько дополнительных файлов, содержащих более подробную информацию о функциональных модулях Navision. Ниже приводится перечень этих файлов и описание их содержимого.

AP.flo содержит блок-схему в формате RFFlow, иллюстрирующую структуру функционального модуля **Accounts Payable**. Файл AP.bmp содержит ту же блок-схему в растровом формате bmp.

Document.flo содержит блок-схему в формате RFFlow, иллюстрирующую структуру типичного объекта класса Document в рамках Navision. Файл document.bmp содержит ту же блок-схему в растровом формате bmp.

Functional Areas.xls представляет собой электронную таблицу в формате Microsoft Excel. Эта таблица содержит один лист для каждого из функциональных модулей Navision, описывающий все объекты, из которых состоят компоненты этого модуля. Информация, приведенная в этом файле, поможет вам быстрее разобраться в архитектуре каждого функционального модуля.

Generic.flo содержит блок-схему в формате RFFlow, иллюстрирующую структуру типичного функционального модуля. Файл generic.bmp содержит ту же блок-схему в растровом формате bmp.

Invoice.flo содержит блок-схему в формате RFFlow, иллюстрирующую структуру приложения Sales Invoicing (Счета-фактуры) в рамках Navision. Файл invoice.bmp содержит ту же блок-схему в растровом формате bmp.

6.3 СТАНДАРТНЫЕ ЗАВИСИМОСТИ В ФУНКЦИОНАЛЬНОМ МОДУЛЕ

Теперь, когда вы уже получили общее представление о функциональных модулях, вам стоит познакомиться с различными связями, существующими в рамках этих модулей. Между объектами каждого функционального модуля существует множество простых зависимостей. Часть наиболее важных зависимостей мы опишем ниже.

Что мы подразумеваем под зависимостью?

Когда мы говорим о зависимостях в рамках функционального модуля, мы имеем ввиду зависимости, существующие между различными таблицами, входящими в этот функциональный модуль. Когда мы говорим, например, что Таблица А зависит от Таблицы В, то мы имеем в виду, что Таблица А содержит поле, обладающее свойством TableRelation и ссылающееся на Таблицу В. Это означает, что значение, содержащееся в этом поле Таблицы А, на самом деле представляет собой значение поля первичного ключа Таблицы В. Более подробную информацию о свойстве TableRelation вы можете найти в курсе *Basic Navision Objects Course*.

Например, пусть таблица **Vendor Ledger Entry (Поставщик Книга Операций)** зависит от таблицы **Vendor (Поставщик)**, и мы знаем, что столбец **No. (Но.)** (обозначающий номер поставщика), является первичным ключом таблицы **Vendor**. Следовательно, в таблице **Vendor Ledger Entry** должно существовать поле **Vendor No. (Поставщик Но.)**, свойство TableRelation которого ссылается на таблицу **Vendor**. Для каждой записи в таблице **Vendor Ledger Entry** поле **Vendor No.** будет содержать значение, соответствующее значению поля **No.** определенной записи в таблице **Vendor**.

Зависимости от основной таблицы

Основная таблица всегда имеет формы **Card (Карточка)** и **List (Список)**. Они играют важную роль в функциональном модуле.

Как журнал, так и учетная книга зависят от основной таблицы.

Кроме того, все вспомогательные таблицы, входящие в этот функциональный модуль, также зависят от основной таблицы.

Зависимости от дополнительных таблиц

Основная таблица, журнал и учетная книга зависят от различных дополнительных таблиц.

Каждая из этих таблиц имеет собственную форму поиска, которая обычно является и формой ввода данных. Доступ к этим таблицам осуществляется нажатием клавиши F6 после перехода в поле, содержащее описание зависимости таблицы.

Таблицы Posting Group (Учетные Группы)

Одной из таких дополнительных таблиц обычно является таблица **Posting Group** данного функционального модуля.

Эта таблица всегда содержит баланс G/L, непосредственно соответствующий основной таблице G/L.

Например, таблица **Vendor Posting Group (Поставщик Учтенные Группы)** содержит балансовый отчет о счетах кредиторов из general ledger (главной бухгалтерской книги). Этот отчет соответствует конкретному поставщик и представляет собой просто сумму всех счетов поставщиков. Можно рассматривать vendor ledger (таблицу учета поставщиков) как вспомогательную таблицу к general ledger, содержащую в себе только учетную информацию о счетах кредиторов.

Для использования этой зависимости пользователь может отключить непосредственную запись учетной информации о счетах кредиторов или любой другой. После этого в учетную информацию о счетах кредиторов записывается только информация, записанная в счет поставщиков. Если это происходит только таким образом, то счета кредиторов на каждую дату должны совпадать с балансом счетов кредиторов для G/L на ту же дату. Связность этих данных обеспечивается таблицей **Posting Group**.

Зависимости от вспомогательных таблиц

От любой основной таблицей также могут зависить и вспомогательные таблицы. Эта связь может быть организована с помощью кнопок меню форм Card или List основной таблицы. Перед отображением этих таблиц основная таблица всегда осуществляет их фильтрацию.

Как правило, но не всегда, вспомогательная таблица зависит также от другой дополнительной таблицы. Если такая зависимость существует,

то получить доступ к этой дополнительной таблице можно таким же образом, как и к любой другой дополнительной таблице, а именно нажав клавишу F6 в поле, содержащем описание зависимостей таблицы.

6.4 Перемещение по функциональному модулю

Нам необходимо научиться переходить от одного объекта к другому, используя стандартный способ перемещения. Изучив информацию, приведенную в этом разделе, вы научитесь получать доступ к данным, содержащимся в функциональном модуле.

Стандартный способ работы с формами основной таблицы

В распоряжении пользователя имеется стандартный способ для работы с любой из основных таблиц (т.е. с таблицами **Customer (Клиент)**, **Vendor (Поставщик)**, **G/L Account (Фин. Счет)** и т.д.). Каждая основная таблица обладает формой Card, с помощью которой в любой момент времени можно просматривать любую, но только одну, из записей этой таблицы и вносить изменения в эту запись. Также каждая основная таблица обладает формой List, которая содержит в себе список всех записей таблицы. Кроме того, каждой основной таблице соответствует учетная таблица. И форма Card, и форма List основной таблицы обладают кнопкой, название которой совпадает с названием основной таблицы. Таким образом, и форма **Customer Card (Клиент Карточка)**, и форма **Customer List (Клиент Список)** содержат кнопку **“Customer” (“Клиент”)**.

Из формы **Customer Card** системы Navision вы всегда можете получить список всех клиентов. Точно также из формы **Vendor Card (Поставщик Карточка)** вы можете получить список всех поставщиков и т.д.

Существует несколько способов открытия списка.

1. Воспользуйтесь кнопкой **List (Список)**, расположенной на панели инструментов.
2. Нажмите клавишу F5.
3. Щелкните кнопку **Customer**, а после этого щелкните **List**.

Вам следует научиться получать список записей основной таблицы всеми из вышеперечисленных способов. Опробуйте их.

Упражнение 1 Перемещение

В главном меню щелкните **Sales & Receivables (Продажи и клиенты)**, а затем щелкните **Customers (Клиенты)**. Откроется карточка клиента.

Нажмите клавишу F5 и выберите запись о любом клиенте. Щелкните OK, чтобы отобразить запись о выбранном клиенте в карточке клиента.

Щелкните кнопку **List (Список)** на панели инструментов и выберите запись о любом клиенте. Щелкните OK, чтобы отобразить запись о выбранном клиенте в карточке клиента.

Щелкните **Customer (Клиент)**, **List (Список)** и выберите запись о любом клиенте. Щелкните OK, чтобы отобразить запись о выбранном клиенте в карточке клиента.

Кроме того, находясь в форме **Customer Card**, вы всегда можете открыть окно **Customer Ledger Entries**. Учетная книга регистрирует все транзакции, совершающиеся с данными заданного клиента, и используется для отображения вычисленных балансов. Аналогичным образом из формы **Vendor Card** вы всегда можете открыть окно **Vendor Ledger Entries** и т.д.

Существует два способа отображения учетных записей.

1 Щелкните пункт меню Customer и выберите Ledger Entries.

2 Нажмите клавиши CTRL+F5.

Вам следует научиться отображать учетные записи обоими вышеперечисленными способами. Опробуйте их:

Упражнение 1 **Перемещение**

В главном меню щелкните **Sales & Receivables (Продажи и клиенты)**, а затем щелкните **Customers (Клиенты)**. Откроется карточка клиента.

Нажмите клавиши CTRL+F5 чтобы просмотреть учетные записи. Нажмите клавишу ESC, чтобы вернуться в карточку клиента.

Чтобы просмотреть учетные записи, щелкните **Customer**, затем **Ledger Entries**. Нажмите клавишу ESC, чтобы вернуться в карточку клиента.

Существует большое количество других стандартных способов

перемещения между формами. Ниже приводится список нескольких из этих способов. Опробуйте каждый из них при работе с несколькими функциональными модулями.

1 **Statistics (Статистика)**: вы можете получить сводные данные практически каждой из основных таблиц, нажав клавишу F9 в форме Card этой основной таблицы.

2 **Journals (Журналы)**: чтобы перейти от записи в журнале к соответствующей карточке (форме Card) основной таблицы воспользуйтесь пунктом меню формы журнала, который называется так же, как и сама основная таблица. Помимо этого, с помощью формы журнала вы можете записывать данные, записывать данные и печатать или же просто печатать тестовый отчет. Доступ к этим функциям осуществляется нажатием на кнопку **Posting (Учет)**.

3 Формы основной таблицы: формы Card и List основной таблицы тесно связаны друг с другом. Как уже упоминалось выше, вы всегда может перейти из формы Card в соответствующую ей форму List. Кроме того, для выбранной записи вы всегда можете перейти из формы List в форму Card. Этот переход осуществляется при нажатии клавиш SHIFT+F5.

4 Формы регистрационных журналов: подменю каждого функционального модуля позволяет вызвать форму регистрационного журнала этого модуля. Из этой формы вы можете перейти к учетным записям, связанным с выбранной в регистрационном журнале записью. Регистрационный журнал содержит в себе информацию обо всех произведенных в этом функциональном модуле операциях записи данных.

Примечание

Так как функциональные модули Navision обладают одинаковой структурой данных, то вам придется потратить меньшее время на изучение этой системы. Если вы изучили один функциональный модуль, то это означает, что вы знакомы с основными принципами работы всех остальных функциональных модулей. Например, формы **Customer Card** и **Vendor Card** очень похожи друг на друга по внешнему виду и выполняемым функциям, несмотря на то, что в их состав входят немного отличающиеся наборы полей. При нажатии клавиши F5 в любой форме Card на экране отобразится список записей основной таблицы данного функционального модуля, а нажатие клавиш CTRL+F5 приведет к отображению учетной таблицы.

Получение дополнительной информации о полях (поиск, детализация и другие функции)

Возможность поиска дополнительной информации является одной из наиболее полезных функциональных особенностей Navision.

Наличие изображения кнопки в поле таблицы говорит о том, что доступна определенная дополнительная информация, относящаяся к данному полю. Щелкнув графический значок или нажав клавишу F6, вы получите доступ к этой информации. Следует иметь в виду, что наличие и тип доступной дополнительной информации зависит от конкретного поля. Ниже вы узнаете, как использовать различные виды таких кнопок для получения дополнительной информации.

1 Щелкните **Sales & Receivables**, затем щелкните **Customers** чтобы открыть форму **Customer Card**.

2 Щелкните поле **Salesperson Code (Код Менеджера)** на закладке **General (Общие свойства)**. Кнопка, находящаяся в этом текстовом поле, называется кнопкой поиска (Lookup). Обратите внимание, что на ней изображена стрелка, указывающая вверх. Щелкните кнопку поиска или нажмите клавишу F6. Откроется форма **Salespeople/Purchasers (Продавцы/покупатели)**. Выберите продавца и щелкните OK, чтобы подтвердить внесение нового значения в **Customer Card**.

Вы видите, что от поля **Salesperson (Менеджер)** можно перейти к таблице, содержащей информацию обо всех доступных продавцах и покупателях. Можно заметить в этом аналогии с зависимостью таблиц, поскольку данное поле “зависит” от информации, содержащейся в другой таблице. Более того, при вводе значения осуществляется проверка целостности данных. Обычно ввод данных в это поле разрешен.

Если на кнопке, находящейся в поле таблицы, изображена стрелка, указывающая вниз, то такая кнопка называется кнопкой детализации (Drill-down). Это поле очень похоже на вычисляемое поле (FlowField), значение которого вычисляется для отображения баланса или другой информации.

1 На закладке **Общее (General)**, щелкните поле **Balance (Баланс)**. Щелкните кнопку детализации или нажмите клавишу F6. После этого на экране отобразится подробный список статей, составляющих вычисленный баланс. Эти данные предназначены только для информационных целей. Обратите внимание, что

поле имеет серый цвет. Это значит, что внесение значения в это поле или изменение этого значения невозможны.

2 Щелкните окно или нажмите клавишу ESC, чтобы закрыть окно.

Также существует третий тип кнопок, называемых кнопками выпадающих меню (Dropdown). Такие кнопки выглядят как треугольники, указывающие вниз. Наличие в поле таблицы такой кнопки говорит о том, что для этого поля существует фиксированный набор опций. Вы можете выбрать одну из этих опций.

1 На закладке **Оплата (Payments)**, щелкните поле **Метод Применения (Application Method)**. Щелкните символ выпадающего меню или нажмите клавишу F6.

2 Нажмите клавишу Esc, чтобы закрыть список опций.

Поля таблиц Navision могут содержать четыре типа кнопок. Кнопки последнего, четвертого, типа называются кнопками **AssistEdit**, на них изображены три точки. Стандартное действие для этих кнопок не определено за исключением того, что при нажатии на такие кнопки открывается другая форма. Вы можете использовать эти кнопки по своему усмотрению.

При нажатии клавиши F6 выполняется одинаковое действие для всех четырех описанных типов кнопок. Тем не менее, изображения, присутствующие на кнопках, различаются. Это позволит вам определить тип поля, с которым вы работаете, а также определить тип дополнительной информации, доступ к которой вы получите, щелкнув кнопку или нажав клавишу F6.

Аналитические измерения

Из любой записи, которая допускает присоединение аналитических измерений, независимо от того, является ли эта запись записью в документе либо записью в основной, журнальной или учетной таблице, пользователь в любой момент может просмотреть присоединенные измерения, нажав клавиши SHIFT+CTRL+D .