Министерство образования
Республики Беларусь

БЕЛОРУССКИЙ НАЦИОНАЛЬНЫЙ
ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

Кафедра «Системы автоматизированного проектирования»

БАЗЫ ДАННЫХ

Лабораторный практикум

Минск
БНТУ
2010

Министерство образования Республики Беларусь
БЕЛОРУССКИЙ НАЦИОНАЛЬНЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

Кафедра «Системы автоматизированного проектирования»

БАЗЫ ДАННЫХ

Лабораторный практикум
для студентов специализации 1-40 01 02-01
«Информационные технологии в производстве и управлении»

Минск
БНТУ
2010

УДК 004.65(076.5)(075.8)
ББК 22.18я7
Б 17

Составители:
канд. техн. наук *В.А. Кочуров, Ю.О. Герман*

Рецензенты:
доцент кафедры САПР БНТУ, канд. техн. наук *В.В. Напрасников,*
доцент кафедры ИТАС БГУИР, канд. техн. наук *О.В. Герман*

Б 17 Базы данных: лабораторный практикум для студентов специализации 1-40 01 02-01 «Информационные технологии в производстве и управлении» / сост.: В.А. Кочуров, Ю.О. Герман. – Минск: БНТУ, 2010. – 90 с.

Настоящий материал предназначен для изучения дисциплин: «Базы данных», «Базы данных и знаний в САПР». Издание содержит шесть лабораторных работ, которые проведут Вас шаг за шагом через процесс создания и использования базы данных на платформе Microsoft SQL Server 2005–2008 и поможет:

- создать необходимые учетные записи;
- определить права доступа;
- создать структуру данных, обладающую ссылочной целостностью и защищенную от несанкционированного доступа;
- заполнить созданные таблицы данными;
- изучить приемы извлечения данных из базы данных.

ISBN 978-985-525-417-2

СОДЕРЖАНИЕ

ЛАБОРАТОРНАЯ РАБОТА №1

ОСНОВЫ РАБОТЫ В MS SQL SERVER

1 Цель работы

1) Изучить интерфейс среды управления SQL сервером – Management Studio.
2) Изучить правила создания таблиц посредством конструктора баз данных в MS SQL Server.
3) Получить начальные практические навыки создания базы данных посредством графического интерфейса среды Management Studio.

2 Задание

1) Создайте учетную запись пользователя MS SQL Server.
2) Создайте новую базу данных.
3) Создайте нового пользователя базы данных.
4) Соединитесь с базой данных под учетной записью пользователя.
5) Создайте структуру базы в соответствии с порядком выполнения работы.
6) Выполните резервное копирование базы данных.

3 Порядок выполнения работы

3.1 Открытие среды SQL Server Management Studio

- В меню **Пуск** последовательно выберите пункты: **Все программы**, **Microsoft SQL Server**, а затем – команду **Среда SQL Server Management Studio**.
- В диалогом окне **Соединение с сервером** подтвердите заданные по умолчанию параметры и нажмите кнопку **Подклю-**

читься. Для соединения необходимо, чтобы поле Имя сервера содержало имя компьютера, на котором установлен SQL Server. Если компонент Database Engine представляет собой именованный экземпляр, то поле «Имя сервера» должно также содержать имя экземпляра в формате <имя_компьютера>\<имя_экземпляра>.

3.2 Соединение с SQL Server

1. Открыть приложение Management Studio путём его выбора из папки Microsoft SQL Server стартового меню (Start Menu → Microsoft SQL Server 2005 → SQL Server Management Studio). В данный момент мы соединяемся с сервером как администратор Windows (рис. 1.1).

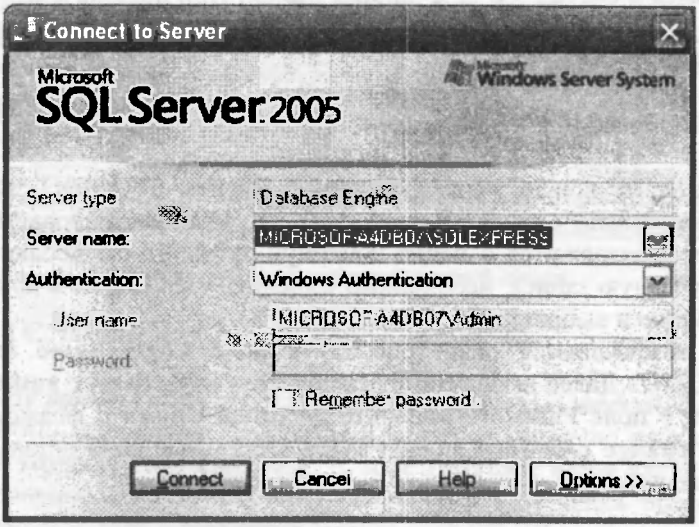

Рис. 1.1. Соединение с SQL Server 2005

2. Выбрать имя сервера (в данном случае это Microsof-A4DB07\SQLEXPRESS).

MS SQL Server использует два режима аутентификации: аутентификацию Windows и аутентификацию MS SQL Server (SQL Server authentication).

Для аутентификации по умолчанию используется аутентификация Windows. **Windows Authentification mode** (Режим аутентификации Windows) — это режим, который использует для подключения к серверу только логины Windows (рис.1.1). В этом случае пользователям нет необходимости вводить какие-то пароли при подключении к SQL Server, если они уже вошли в сеть Windows.

Для использования **аутентификации SQL Server** вначале необходимо создать учетную запись в самом SQL Server. В отличие от логинов Windows, логины SQL Server – это самостоятельные учетные записи со своими именами и паролями, информация о которых хранится в системной базе данных master. При подключении к серверу при помощи логина SQL Server вам придется явно указать имя логина и пароль.

3.3 Создание учетной записи

Создайте новую учетную запись с именем STUDENT. В Management Studio список учетных записей, сконфигурированных на сервере, содержится в папке \Security\Logins. Чтобы добавить новую учетную запись, необходимо выделить узел **Logins** в контекстном меню и выбрать пункт **New Login**.

В открывшемся окне (рис.1.2) в поле **Login name** введите *STUDENT*. Далее выберите переключатель **SQL Server authentication** и в поле **Password** наберите *goodstud*. Снимите флажок **User must change password at next login**. Остальные поля оставьте без изменений.

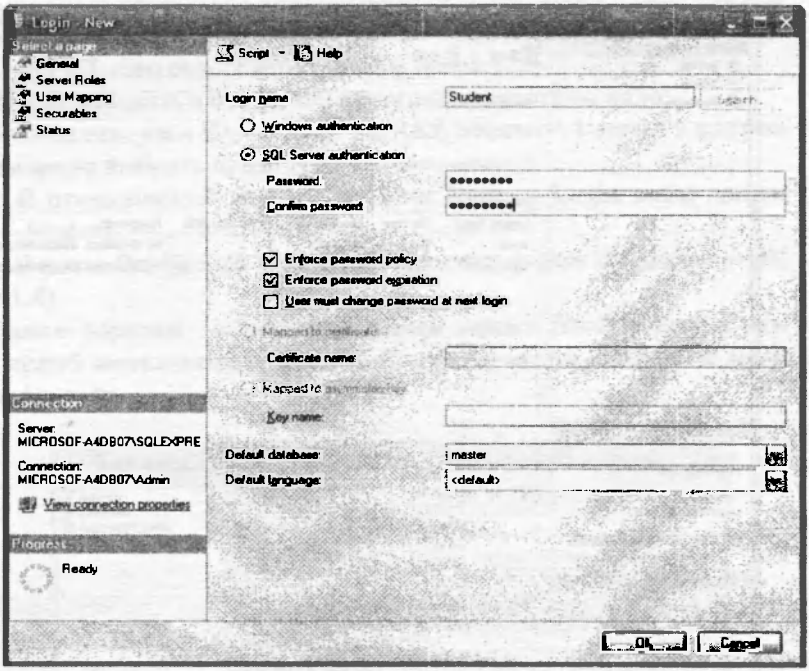

Рис. 1.2. Создание новой учетной записи

3.4 Создание базы данных

Базу данных может создавать только пользователь с правами администратора. Выберите в контекстном меню папки **Databases** команду **New Database**. В поле **Database name** введите имя создаваемой базы данных (БД) – Borei (рис. 1.3). Здесь также можно изменить путь сохранения создаваемой БД. *Обратите внимание, что создаётся пустая база данных (контейнер).*

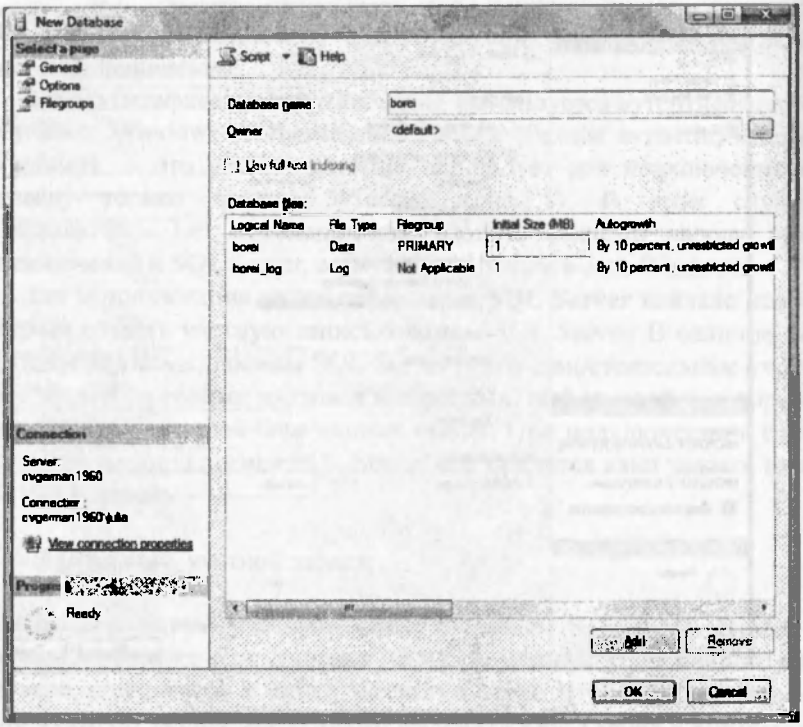

Рис. 1.3. Создание новой базы данных

После создания БД окно **Object Explorer** обновится. В ветке **Databases** появится новая база данных BOREI. Эта база данных принадлежит пользователю по имени **sysadmin** (системный администратор).

3.5 Создание нового пользователя

Создание пользователя в SQL Server 2005 осуществляется посредством создания учетной записи. ***Пользователи базы данных*** – это специальные объекты, которые создаются на уровне базы дан-

ных и используются для предоставления разрешений в базе данных (на таблицы, представления, хранимые процедуры). Логины и пользователи БД - это совершенно разные объекты.

Чтобы добавить в базу Borei нового пользователя, надо:

1. выделить **узел \Databases\ BOREI\ Security\ Users** и в контекстном меню выбрать пункт **New User**.

2. В открывшемся окне в поля **User Name** и **Login name** ввести имя – Student.

3. В списке **Database role membership** установить флажок **db_owner**→OK (рис.1.4).

Таким образом, на основе учетной записи Student будет создан новый пользователь для БД Borei с правами владельца этой базы данных.

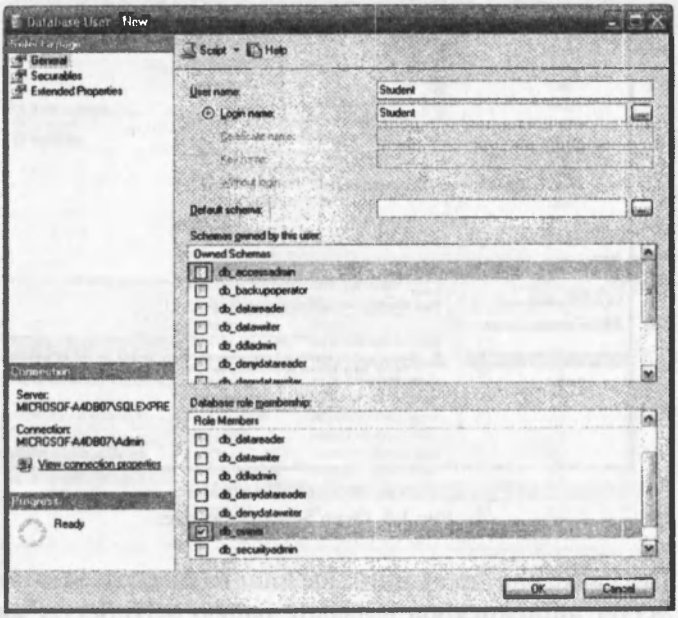

Рис. 1.4. Добавление в БД нового пользователя

3.6 Соединение с базой данных

Чтобы пользователь Student мог подсоединиться к созданной базе данных с целью создания её структуры, следует выполнить несколько подготовительных действий.

1. Войдите в окно свойств SQL Server: в окне Object Explorer выберите корневой элемент (название машины, на которой запущен сервер) и вызовите для него контекстное меню. Выберите пункт Properties. Откроется окно Server Properties (рис. 1.5).

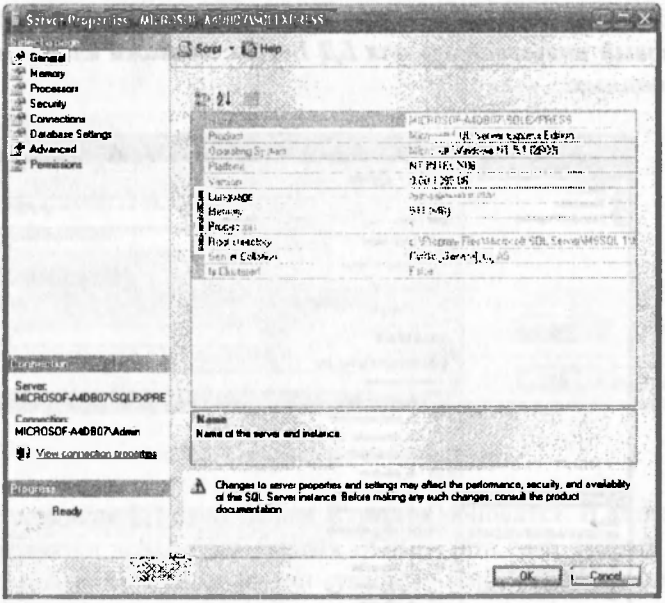

Рис.1.5. Окно Server Properties

2. Выберите способ аутентификации: на вкладке **Security** в разделе **Server authentication** выберите опцию **SQL Server and Windows Authentication mode**.

3. На вкладке **Permissions** нажмите кнопку **Add**. Откроется окно **Select Logins or Roles**→ Browse.

4. Определите учётную запись, под которой в последующем мы будем заходить на сервер: в открывшемся окне Browse for Objects в списке **Matching objects** выберите пункт [Student], поставьте возле него галочку → OK. В окне **Select Logins or Roles** в поле **Enter the object names to select** появится текст [Student] → Ok. Теперь в окне **Server Properties** на вкладке **Permissions** в списке **Logins or roles** появился пункт Student (в конце списка). Выберите его.

5. Устанавливаем права доступа к серверу для выбранной учётной записи: в списке Explicit permissions for Student ставим птички в колонке Grant возле пунктов Authenticate server и Connect SQL (рис. 1.6).

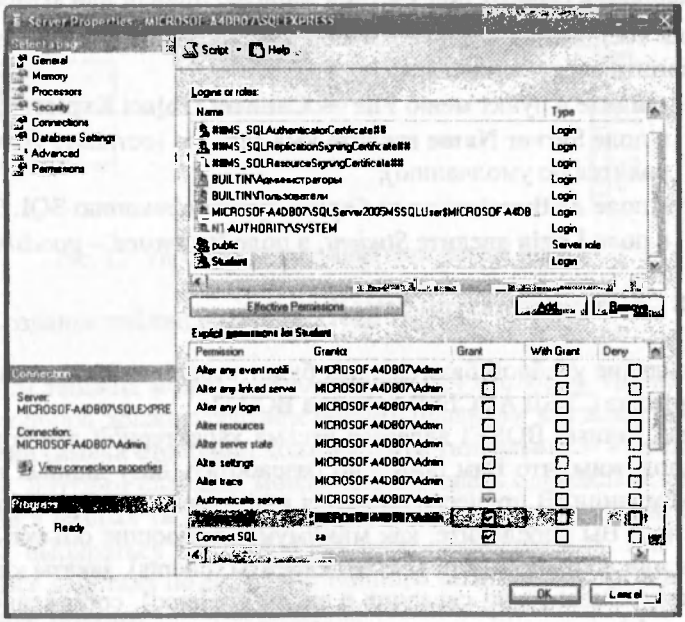

Рис. 1.6. Установление прав доступа к серверу

6. Заканчиваем подготовительные действия: нажимаем ОК в окне Server Properties. Появится предупреждение о том, что сервер необходимо перезагрузить, чтобы изменения вступили в силу. Нажимаем ОК.

7. Для перезагрузки сервера в окне **Object Explorer** в корневом элементе вызываем контекстное меню и выбираем пункт **Restart** → Yes. Будет произведен перезапуск службы сервера.

***Примечание**. Выполнять пункты 1-2 надо один раз. Пункты 3-7 необходимо выполнять для каждой учётной записи (нового логина).*

3.7 Подключение к базе данных

Чтобы выполнять действия, описанные ниже, нужно быть подключенным к серверу под учетной записью Student или sysadmin.

Для соединения с сервером под учетной записью Student, надо выполнить следующие шаги:

- зайдите в пункт меню **File → Connect Object Explorer**;
- в поле **Server Name** введите имя сервера (оставьте то, что выводится по умолчанию);
- в поле **Authentication** выберите аутентификацию SQL Server;
- в поле **Login** введите *Student*, в поле **Password** – *goodstud*.

3.8 Создание структуры базы данных

Создание учебной базы данных будет выполняться на основе известного из СУБД ACCESS примера BOREI.

База данных BOREI является весьма характерной для торговли. Предположим, что Вам поручено разработать базу данных некоторой компании. В процессе создания информационной модели (data modeling) Вы определите, как минимум, следующие объекты (сущности или абстрактные объекты): клиенты (clients), заказы клиентов (orders), заказано или сведения о заказе (ordered), собственно товары (goods), сотрудники компании (employees), доставка (deliveries), а также поставщиков сырья (suppliers), типы или категории товаров (types). При нашем дальнейшем изучении можно увидеть, что база

данных BOREI содержит восемь таблиц в точности соответствующих вышеприведённым объектам (рис.1.7).

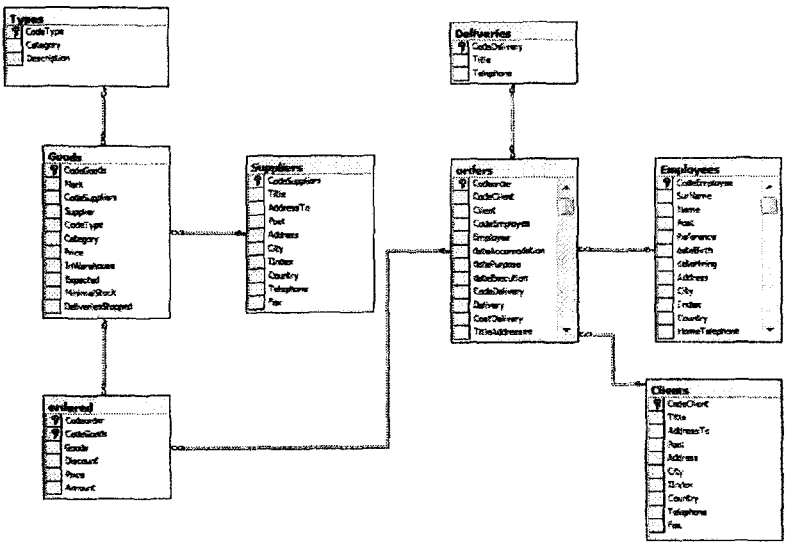

Рис. 1.7. Упрощённая диаграмма базы данных BOREI

3.9 Создание таблиц в конструкторе таблиц

Каждая таблица может содержать до 1024 столбцов. Имена таблиц и столбцов должны соответствовать правилам для идентификаторов: они (имена столбцов) должны быть уникальными в пределах таблицы, другими словами, можно использовать одинаковые имена столбцов в разных таблицах одной БД. Назначаемые данным типы должны выбираться из типов данных, используемых MS SQL Server. Все действия по созданию таблицы можно выполнить **в конструкторе таблиц:**

1. Щелкните правой кнопкой узел таблиц базы данных в **Обозревателе объектов** (Object Explorer) и нажмите кнопку **Создать**

таблицу (New Table). В появившемся окне следует ввести следующую информацию:

- **Column Name** – имя поля;
- **Data Type** – тип поля (см. Конспект лекций п. 2.3);
- **Allow Nulls** – разрешение не вводить данные в столбец. Значение NULL обозначает факт отсутствия какого-либо значения.

2. Заносим в поля значения, как показано на рисунке 1.8.

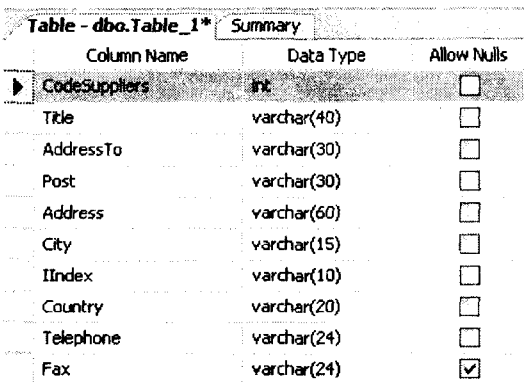

Рис. 1.8. Создание таблицы Suppliers (Поставщики)

3. Сделаем поле CodeSuppliers ключевым.

Для создания первичного ключа:

- в конструкторе таблиц щелкните селектор строк для столбца базы данных, который необходимо определить в качестве первичного ключа. Чтобы выделить несколько столбцов, нажмите и удерживайте клавишу CTRL и щелкните селекторы строк для остальных столбцов.
- Щелкните правой кнопкой мыши селектор строк столбца и выберите команду **Задать первичный ключ**. Будет автоматически создан индекс первичного ключа с именем, состоящим из «PK_» и имени таблицы; его можно найти в диалоговом окне Индексы и ключи.

Для изменения первичного ключа:

- откройте в конструкторе таблиц таблицу, чей первичный ключ необходимо изменить, правой кнопкой мыши щелкните **Конструктор таблиц** и выберите пункт Индексы/Ключи в контекстном меню.
- В диалоговом окне Индексы/Ключи выберите индекс первичного ключа из списка Выберите первичный/уникальный ключ или индекс.
- *Выполните действие*, руководствуясь таблицей 1.1.

Таблица 1.1 - Изменение первичного ключа

Действие	Процедура выполнения
Переименование первичного ключа	Введите новое имя в поле **Имя**. Убедитесь, что новое имя не совпадает с именами в списке Выбранный первичный/уникальный ключ или индекс.
Установка параметра кластеризации	Выберите **Создать** как CLUSTERED, и из раскрывающегося списка выберите нужный параметр. В таблице может существовать только один кластеризованный индекс. Если этот параметр недоступен для выбранного индекса, то сначала снимите этот флажок в существующем кластеризованном индексе.
Определение коэффициента заполнения	Разверните категорию **Определение заполнения** и введите целое число от 0 до 100 в поле Коэффициент заполнения.
Изменение порядка столбцов	Выберите **Столбцы** и нажмите троеточие (...) справа от свойства. В диалоговом окне Столбцы индекса удалите столбцы из первичного ключа. Затем снова добавьте эти столбцы в необходимом порядке. Чтобы удалить столбец из ключа, просто удалите имя столбца из списка имен Столбец.

4. В нижнем окне (Column properties) можно указать следующую информацию:

- **Description** – описание столбца (вводится произвольная информация);
- **Default Value** – значение в столбце, вводимое по умолчанию (если пользователь не ввел данные в столбец);
- **Identity** – свойство IDENTITY обычно используется для автоматического присвоения уникальных идентификационных номеров или первичных ключей.

Примечание. *Свойство IDENTITY, может использоваться только с целочисленными значениями.*

- **Identity Seed** – начальное значение счетчика записей, которое присваивается первой записи, загружаемой в таблицу;
- **Identity Increment** – значение приращения, которое прибавляется к значению идентификатора предыдущей загруженной строки. Необходимо указывать либо оба аргумента (и seed, и increment), либо не указывать ни одного из них. Если ничего не указано, применяются значения по умолчанию (1,1).

5. Набранную таблицу следует сохранить, используя для этого кнопку  и введя имя Suppliers.

Создание таблицы "клиенты" (Clients)

Создадим таблицу клиенты в соответствии с вышеприведённым сценарием и нижеследующими данными:

```
CodeClient          int      NOT NULL,
Title               VARCHAR(40) NOT NULL,
AddressTo           VARCHAR(30) NOT NULL,
Post                VARCHAR(30) NOT NULL,
Address             VARCHAR(60) NOT NULL,
City                VARCHAR(15) NOT NULL,
IIndex              VARCHAR(10),
Country             VARCHAR(20) NOT NULL,
Telephone           VARCHAR(24) NOT NULL,
Fax                 VARCHAR(24)
```

Создадим первичный ключ по полю **CodeClient**. Для этого выде-

лим его и в контекстном меню выберем пункт **Set Primary key**.

Сохраним описание таблицы посредством кнопки  и введя имя **Clients**.

Создание таблицы "заказы" (Orders)

Аналогичным образом создадим таблицу заказы в соответствии с нижеприведёнными данными. Зададим для неё первичный ключ по полю **CodeOrder**.

```
CodeOrder           int NOT NULL,
CodeClient          int NOT NULL,
Client              VARCHAR(40),
CodeEmployee        int NOT NULL,
Employee            VARCHAR(30),
DateAccomodation    datetime,
DatePurpose         datetime,
DateExecution       datetime,
CodeDelivery        int NOT NULL,
Delivery            VARCHAR(40),
CostDelivery        numeric(10,3),
TitleAddressee      VARCHAR(40) NOT NULL,
AddressAddressee    VARCHAR(60) NOT NULL,
CityAddressee       VARCHAR(15) NOT NULL,
IndexAddressee      VARCHAR(10) NOT NULL,
CountryAddressee    VARCHAR(20) NOT NULL
```

Создание таблицы " товары " (Goods)

Теперь можно создать таблицу "товары" (goods) с первичным ключом по полю **CodeGoods**.

```
CodeGoods           int NOT NULL,
Mark                VARCHAR(40) NOT NULL,
CodeSuppliers       int NOT NULL,
Supplier            VARCHAR(40),
CodeType            int NOT NULL,
Category            VARCHAR(30),
Price               numeric(10,3),
InWarehouse         int,
Expected            int,
MinimalStock        int,
DeliveriesStopped   varchar(3)
```

Создание таблицы "Типы или категории" (Types)

Подобным образом создадим таблицу "Типы или категории" согласно нижеследующим данным с первичным ключом **CodeType**.

```
CodeType        int NOT NULL,
Category        VARCHAR(30) NOT NULL,
Description     text
```

Создание таблицы "заказано" (Ordered)

Таким же образом создадим таблицу "заказано" в соответствии с нижеприведёнными данными. Здесь первичный ключ *составной*, т.к. состоит из двух полей: **CodeOrder, CodeGoods**.

```
CodeOrder       int NOT NULL,
CodeGoods       int NOT NULL,
Goods           VARCHAR(40),
Discount        numeric(5,4),
Price           int
```

Создание таблицы "сотрудники" (Employees)

Согласно следующим данным создадим таблицу сотрудников компании с первичным ключом **CodeEmployee**.

```
CodeEmployee    int NOT NULL,
SurName         VARCHAR(20) NOT NULL,
Name            VARCHAR(10) NOT NULL,
Post            VARCHAR(30) NOT NULL,
Reference       VARCHAR(25) NOT NULL,
DateBirth       datetime,
DateHiring      datetime,
Address         VARCHAR(60) NOT NULL,
City            VARCHAR(15) NOT NULL,
IIndex          VARCHAR(10) NOT NULL,
Country         VARCHAR(20) NOT NULL,
HomeTelephone   VARCHAR(24) NOT NULL,
Additional      VARCHAR(24) NOT NULL,
Photo           text,
Note            text,
Submits         VARCHAR(30)
```

Создание таблицы "доставка" (Deliveries)

Также по нижеследующим данным создадим таблицу доставка с первичным ключом **CodeDelivery**.

```
CodeDelivery        int NOT NULL,
Title               VARCHAR(40) NOT NULL,
Telephone           VARCHAR(24) NOT NULL
```

3.10 Резервное копирование базы данных

Данные, хранящиеся в базах данных, являются стратегической ценностью любой компании, поэтому их сохранность – наивысший приоритет!

1. Выберем в контекстном меню базы данных BOREI пункт Tasks → Back Up...

2. В открывшемся диалоге Back Up Database – BOREI (рис. 1.9) установим некоторые параметры:

- Database — BOREI (это база данных для резервного копирования);
- Backup type — Full (тип резервной копии — полный; для первого резервного копирования применяется только такой; если резервная копия уже имеется и вы не желаете создавать заново полную копию, а внести изменения в старую, выберите пункт Differential);
- Backup component — Database;
- Name — поле для ввода имени резервной копии; можно оставить по умолчанию;
- Back up to: — Disk.

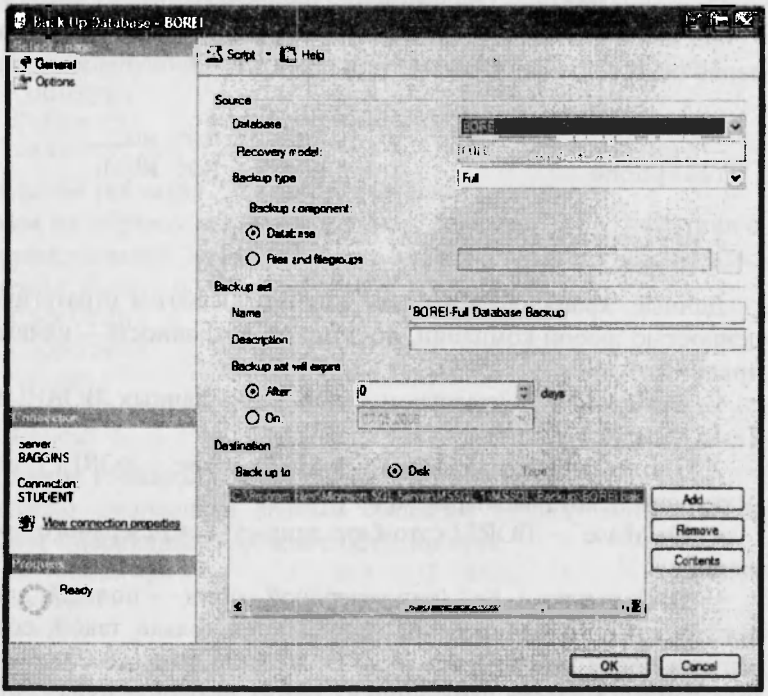

Рис. 1.9. Диалоговое окно Back Up Database - BOREI

3. Нажмите кнопку **Add...** в этом диалоге. Откроется диалоговое окно Select Backup Destination (рис.1.10). В нем в поле **File name** введите имя файла, в который будет выполнено резервное копирование (у нас это «C:\BOREI\BACKUPS\backup1.bak»).

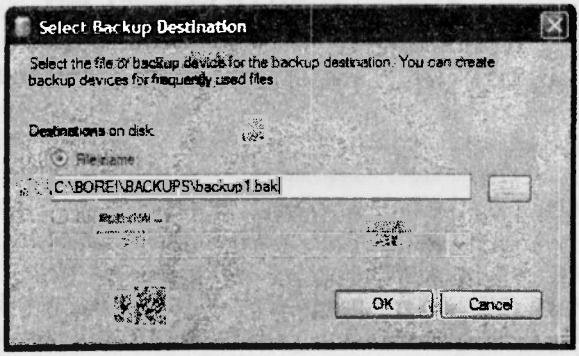

Рис. 1.10. Диалоговое окно Select Backup Destination

4. Нажмите OK в диалоговом окне Select Backup Destination. Оно закроется, а в списке Destination окна Back Up Database – BOREI появится строка «C:\BOREI\BACKUPS\backup1.bak» (рис.1.9). Выберите ее и нажмите кнопку OK. Произойдет резервное копирование по указанному пути, после чего появится диалог с текстом «The backup of database 'BOREI' completed successfully». Нажмите OK и в нем.

4 Контрольные вопросы

1) Расскажите о назначении среды Management Studio.
2) Как открыть среду Management Studio?
3) Что такое пользователь базы данных и как он создается?
4) Как создать новую базу данных в SQL Server?
5) Как осуществляется соединение пользователя с базой данных?
6) Расскажите о порядке создания таблицы базы данных?
7) Расскажите о первичном индексе SQL Server.
8) Как осуществляется резервное копирование базы данных?

ЛАБОРАТОРНАЯ РАБОТА № 2

АРХИТЕКТУРА БАЗ ДАННЫХ СЕРВЕРА SQL

1 Цель работы

1) Изучить приёмы работы с диаграммами баз данных.
2) Изучить приёмы изменения таблиц и их свойств.
3) Изучить правила создания индексов.
4) Изучить правила создания отношений между таблицами.
5) Изучить приёмы поддержания ссылочной целостности базы данных.
6) Изучить правила создания ограничений.

2 Задание

1. Построить начальную версию диаграммы базы данных Борей.
2. Используя визуальные инструменты конструктора таблиц, выполнить предлагаемые примеры по созданию индексов.
3. Используя визуальные инструменты конструктора баз данных:
   3.1. установить отношение между родительской таблицей **Types** на основе её первичного ключа **CodeType** и дочерней таблицей **Goods** посредством её внешнего ключа **CodeType** путём создания ограничения внешнего ключа для таблицы Types;
   3.2. установить отношение между родительской таблицей **Suppliers** на основе её первичного ключа **CodeSuppliers** и дочерней таблицей **Goods** посредством её внешнего ключа **CodeSupliers** путём создания ограничения внешнего ключа для таблицы Goods;
   3.3. установить отношение между родительской таблицей **Clients** на основе её первичного ключа **CodeClient** и до-

черней таблицей **Orders** посредством её внешнего ключа **CodeClient** путём создания ограничения внешнего ключа для таблицы Orders;

3.4. установить отношение между родительской таблицей **Employees** на основе её первичного ключа **CodeEmployee** и дочерней таблицей **Orders** посредством её внешнего ключа **CodeEmployee** путём создания ограничения внешнего ключа для таблицы Orders.

4. Используя конструктор таблиц:

4.1. установить отношение между родительской таблицей **Deliveries** на основе её первичного ключа **CodeDelivery** и дочерней таблицей **Orders** посредством её внешнего ключа **CodeDelivery** путём создания ограничения внешнего ключа для таблицы Orders;

4.2. установить отношение между родительской таблицей **Orders** на основе её первичного ключа **CodeOrder** и дочерней таблицей **Ordered** посредством её внешнего ключа **CodeOrder** путём создания ограничения внешнего ключа для таблицы **Ordered;**

4.3. установить отношение между родительской таблицей **Goods** на основе её первичного ключа **CodeGoods** и дочерней таблицей **Ordered** посредством её внешнего ключа **CodeGoods** путём создания ограничения внешнего ключа для таблицы **Ordered.**

5. Добавить проверочные ограничения для соответствующих полей таблицы Goods:
   - InWarehouse >= 0;
   - Expected >= 0;
   - MinimalStock >= 0;
   - (DeliveriesStopped = 'NO') OR (DeliveriesStopped = 'YES').
6. Выполнить резервное копирование базы данных.

3 Порядок выполнения работы

3.1 Создание диаграммы

Для создания новой диаграммы базы данных:

1. в обозревателе объектов щелкните правой кнопкой мыши папку Диаграммы баз данных или любую диаграмму в этой папке.
2. Выберите в контекстном меню значок **Создать диаграмму базы данных**. Появится диалоговое окно **Добавить таблицу**.
3. Выберите последовательно из списка **Таблицы** все таблицы базы данных Борей, созданные в предыдущей лабораторной работе, и, выбирая опцию **Добавить,** перенесите их на создаваемую диаграмму. Эти таблицы отобразятся в графическом виде в новой диаграмме базы данных.
4. Меню **Диаграммы баз данных** будет добавлено в главное меню, и при этом откроется область конструктора.
5. Продолжайте добавлять и удалять таблицы, изменять существующие таблицы и изменять связи между таблицами до завершения новой диаграммы базы данных.

Для открытия существующей диаграммы базы данных:

1. в обозревателе объектов раскройте папку Диаграммы баз данных.
2. Дважды щелкните имя схемы базы данных, которую нужно открыть,

-или-

3. щелкните правой кнопкой мыши имя схемы базы данных, которую нужно открыть, а затем выберите **Создать диаграмму базы данных**.
4. Схема базы данных открывается в конструкторе схем базы данных, где можно изменить схему.

Примечание. *Только владелец диаграммы или член роли базы данных db_owner может открыть диаграмму.*

Для сохранения диаграммы базы данных:

1. в меню **Файл** выберите **Сохранить** <diagram>.
2. Если это новая диаграмма, которую еще ни разу не сохраняли, откроется диалоговое окно **Выбор имени** Введите имя диаграммы.
3. Если были выполнены изменения в таблицах существующей диаграммы, откроется диалоговое окно **Сохранить**, в котором отображается список изменений, которые будут сохранены в базе данных при сохранении диаграммы.
4. Нажмите кнопку **Да** (или **ОК**, если это новая диаграмма) для обновления базы в соответствии с диаграммой.

При сохранении диаграммы базы данных будут сохранены все изменения, включая изменения, выполненные в таблицах, столбцах и других ее объектах.

Если нет необходимости сохранять все изменения, внесенные в диаграмму базы данных, можно сохранить определенную таблицу или набор таблиц:

1. в диаграмме базы данных выберите таблицы для сохранения.
2. В меню **Файл** выберите **Сохранить выделенное**. В диалоговом окне **Сохранить** откроется список таблиц, которые будут обновлены в базе данных при сохранении выделенных таблиц.
3. Выберите **Сохранить текстовый файл**, чтобы перед продолжением сохранить список таблиц в текстовом файле в каталоге проекта.
4. В диалоговом окне **Сохранить** проверьте список таблиц и нажмите кнопку **Да**, чтобы их сохранить.

Примечание. *Список таблиц может содержать и другие таблицы, кроме выделенных. Например, если изменить тип данных столбца, связанного с другой таблицей, в этот список будут включены обе таблицы.*

Для вставки новой таблицы в диаграмму:

1. убедитесь, что осуществлено подключение к базе данных, в которой требуется создать таблицу.

2. Нажмите на панели инструментов кнопку **Создать таблицу,**

   -или-

   щелкните правой кнопкой мыши диаграмму и выберите пункт **Создать таблицу**.
3. Измените или сохраните имя таблицы, назначенное системой, в диалоговом окне **Выбор имени** и нажмите кнопку **ОК**. Откроется стандартный конструктор таблиц, в котором уже известным нам образом можно определить свойства столбцов и ограничений.
4. При сохранении диаграммы таблица будет создана в базе данных.

Для добавления существующей таблицы в диаграмму:

1. убедитесь, что осуществлено подключение к той базе данных, таблицы которой требуется изменить.
2. Выберите таблицу в папке **Таблицы**.
3. Перетащите таблицу в диаграмму базы данных.
4. Отпустите кнопку мыши. Таблица будет вставлена в диаграмму. Если вставляемая таблица имеет связи с уже существующими на диаграмме таблицами, то они будут автоматически отображены.

Для создания отношения «многие ко многим» между таблицами:

1. добавьте таблицы, которые необходимо связать отношением «многие ко многим» в диаграмму базы данных.
2. Создайте третью таблицу, щелкнув диаграмму правой кнопкой мыши и выбрав **Создать таблицу**. Эта таблица станет связующей.
3. В диалоговом окне **Выбор имени** измените имя, назначенное системой. Например, связующую таблицу для таблиц titles и authors можно назвать titleauthors.
4. Скопируйте столбцы первичных ключей обеих таблиц в связующую таблицу. В эту таблицу можно добавить другие столбцы, как в любую другую таблицу.
5. Создайте первичный ключ в связующей таблице так, чтобы

он содержал все столбцы первичных ключей исходных таблиц.

6. Определите отношение «один ко многим» между каждой из первоначальных таблиц и связующей таблицей. Связующая таблица должна находиться на стороне «многих» обоих отношений.

Для создания рефлексивной связи:

1. в диаграмме базы данных щелкните переключатель строк для столбца базы данных, который необходимо связать с другим столбцом, и перетаскивайте указатель за пределы таблицы, пока не появится линия.
2. Перетащите линию назад к выбранной таблице.
3. Отпустите кнопку мыши. Появится диалоговое окно Таблицы и столбцы.
4. Выберите столбец внешнего ключа и таблицу первичного ключа, с которой необходимо установить связь.
5. Дважды нажмите кнопку *ОК*, чтобы создать связь.

Можно скопировать таблицу из одной диаграммы базы данных в другую в той же самой базе данных. Копирование таблицы из одной диаграммы базы данных в другую диаграмму добавляет ссылку на таблицу во второй диаграмме. При этом таблица не будет продублирована в базе данных. Например, при копировании таблицы authors из одной диаграммы базы данных в другую, каждая диаграмма будет ссылаться на одну и ту же таблицу authors в базе данных.

Для копирования таблицы из одной диаграммы в другую:

1. убедитесь в наличии соединения с базой данных, таблицу которой необходимо скопировать.
2. Откройте исходные и целевые диаграммы базы данных и в исходной диаграмме выберите таблицу, которую необходимо скопировать в целевую диаграмму.
3. Нажмите кнопку **Копировать** на панели инструментов или в меню **Схема базы данных** (или меню **Правка**) сделайте выбор ***Копировать диаграмму в буфер обмена***. Это действие

помещает выбранное определение таблицы в буфер обмена.

4. Переключитесь к целевой диаграмме. Эта диаграмма должна быть в той же самой базе данных, где и исходная диаграмма.
5. Нажмите кнопку **Вставить** на панели инструментов или меню **Правка** выберите **Вставить**. Содержимое буфера обмена появится в новом месте и останется выделенным, пока не будет выполнен щелчок где-либо в другом месте. Если существуют связи между выбранными таблицами и другими таблицами в целевой диаграмме, линии связи будут нарисованы автоматически.

При редактировании таблицы в любой диаграмме изменения будут отражены в обеих диаграммах. Точно так же при сохранении таблицы в любой диаграмме таблица больше не считается измененной в любой диаграмме.

3.2 Создание индексов

Индексы предназначены для ускорения доступа к данным в таблице базы данных. Индекс можно создать, выбрав один или несколько столбцов таблицы, по которым необходимо выполнять поиск. Индексом можно будет пользоваться сразу же после сохранения таблицы.

Для создания индекса:

1. в обозревателе объектов щелкните правой кнопкой мыши таблицу, для которой хотите создать полнотекстовый индекс и выберите **Проект** (**Изменить** в версии с пакетом обновления 1 (SP1) или более ранней версии). Таблица откроется в **Конструкторе таблиц**.
2. В меню **Конструктора таблиц** выберите пункт **Индексы и Ключи**.
3. В диалоговом окне **Индексы и Ключи** нажмите **Добавить**.
4. Выберите **новый индекс** в списке **Выбранный первичный/уникальный ключ или индекс** и определите его свойства в таблице справа.

5. При необходимости укажите дополнительные параметры индекса и нажмите кнопку **Закрыть**. Индекс создается в базе данных при сохранении таблицы.

Для создания уникального индекса:

- находясь в сетке, щелкните **Тип**.
- Выберите **Индекс** в раскрывающемся списке справа от свойства.
- В списке **Имя столбца** выберите столбцы, которые необходимо проиндексировать. Выбрать можно до 16 столбцов. Для оптимальной производительности следует выбирать только один или два столбца на индекс. Для каждого выбранного столбца укажите, будут ли значения данного столбца располагаться в индексе в возрастающем или убывающем порядке.
- Находясь в сетке, щелкните **Уникальный**.
- Выберите **Да** в раскрывающемся списке справа от свойства.
- Выберите параметр **Пропустить повторяющиеся ключи**, чтобы данные, которые создает дублирующийся ключ в уникальном индексе (при помощи инструкции INSERT), были пропущены.

Для создания кластеризованного индекса:

- выберите в сетке **Создать как кластеризованный** и из раскрывающегося списка справа от свойства выберите **Да**.

Для изменения свойств индекса:

- находясь в сетке диалогового окна **Индексы и Ключи** выберите индекс из списка **Выбранный первичный / уникальный ключ или индекс**.
- Измените свойства в сетке. Изменения будут сохранены в базе данных при сохранении таблицы.

Переименование индекса.

Новым индексам автоматически присваиваются системные имена в зависимости от имени таблицы базы данных. При создании нескольких индексов таблицы к именам индексов добавляются «_1», «_2» и т.д. Индекс можно переименовать при условии, что его имя будет уникальным в пределах таблицы.

Примечание. *При создании первичного ключа или ограничения уникальности таблицы автоматически создается индекс с таким же именем, что и у ограничения. Поскольку имена индексов должны быть уникальны в пределах таблицы, нельзя создавать или переименовывать индекс так, чтобы его имя совпадало с именем первичного ключа или ограничения уникальности.*

Для переименования индекса:

- Находясь в сетке диалогового окна **Индексы и Ключи,** выберите индекс из списка **Выбранный первичный / уникальный ключ или индекс**.
- В сетке выберите **Имя** и введите новое имя в текстовое поле.
- Изменения будут сохранены в базе данных при сохранении таблицы.

Удаление индекса

Индексы могут уменьшить производительность инструкций INSERT, UPDATE и DELETE. Индекс можно удалить, если он уменьшает общую производительность или больше не нужен.

Для удаления индекса:

- в диалоговом окне Индексы и Ключи выберите индекс, который хотите удалить.
- Нажмите кнопку **Удалить.** Индекс удаляется из базы данных при сохранении таблицы.

Примечание. *Если индекс был удален в* ***конструкторе таблиц****, но не нужно удалять его из базы данных, можно закрыть таблицы без сохранения. Это также отменит все другие изменения с момента последнего сохранения таблицы.*

➧**Пример 1. Создание индекса Namex.** Создайте в конструкторе таблиц индекс для таблицы Employees, определяемый следующей инструкцией:

CREATE INDEX Namex ON Employees (Surname, Name).

Эта инструкция определяет индекс с именем Namex для таблицы Employees на основе полей Surname и Name.

➧**Пример 2. Создание уникального индекса.** Создайте в кон-

структоре таблиц уникальный индекс с именем Goodsx для таблицы Goods, определяемый следующей инструкцией:

CREATE UNIQUE INDEX Goodsx ON Goods (Mark).

➧**Пример 3. Создание индекса с убывающим порядком.** По умолчанию SQL сохраняет ключи индекса в порядке возрастания. Для изменения порядка на *убывающий* нужно включить в определение индекса ключевое слово **DESC**. Создайте в конструкторе таблиц уникальный индекс с именем Amountx для таблицы Ordered, определяемый следующей инструкцией:

CREATE INDEX Amountx ON Ordered (codeorder DESC, amount DESC).

➧**Пример 4. Модификация индексов.** Для изменения определения индекса следует его удалить оператором, и создать заново.

3.3 Установление отношений между таблицами базы данных Борей

Можно создавать связи между столбцами разных таблиц непосредственно в конструкторе диаграмм, перетаскивая столбцы между таблицами. Для выполнения п.п. 3.1 – 3.4 Задания настоящей лабораторной работы (***создание ограничения FOREIGN KEY посредством конструктора баз данных)*** следует выполнить нижеследующую последовательность действий:

1. в конструкторе баз данных щелкните селектор строк для одного или более столбцов базы данных, которые необходимо связать со столбцом в другой таблице.

   Примечание. *Столбцы, которые выбираются для внешнего ключа, должны иметь одинаковый тип данных с первичными столбцами, которым они соответствуют. Каждый ключ должен содержать одинаковое число столбцов. Например, если первичный ключ на первичной стороне связи состоит из двух столбцов, необходимо сопоставить каждому из этих столбцов столбец таблицы, который будет*

входить во внешний ключ на другой стороне связи.

2. Перетащите выбранный столбец (столбцы) в связанную таблицу.
3. Отобразятся два диалоговых окна: **Связи внешнего ключа** и **Таблицы и столбцы**, второе отображается на переднем плане.
4. **Имя связи** устанавливается системой в формате FK_ *таблица первичного ключа _таблица внешнего ключа*. Можно изменить это значение.
5. Убедитесь, что **Таблица первичного ключа** правильно задает таблицу.
6. Сетка содержит локальные столбцы и соответствующие им внешние столбцы. Можно добавить или удалить столбцы таблицы, либо изменить сопоставления.
7. Нажмите кнопку **ОК**. Откроется диалоговое окно **Связи внешнего ключа**. **Выбранная связь** отображает созданную связь.
8. Измените свойства связи в сетке.
9. Нажмите кнопку OK, чтобы создать связь. Конструктор баз данных отобразит связь между выбранными столбцами.

Для выполнения п.п. 4.1 – 4.3 Задания настоящей лабораторной работы (***создание ограничения FOREIGN KEY посредством конструктора таблиц)*** выполним следующую последовательность действий:

1. в **обозревателе объектов** щелкните правой кнопкой мыши таблицу на стороне внешнего ключа для связи и выберите **Проект** (**Изменить** в версии с пакетом обновления 1 (SP1) или более ранней версии). Таблица откроется в конструкторе таблиц.
2. В меню конструктора таблиц выберите пункт **Отношения**.
3. В диалоговом окне **Выбранный элемент Отношение** щелкните **Добавить**. Связь появится в списке этого окна с установленным системой именем, в формате FK_<tablename>_<tablename>, где tablename является именем таблицы внешнего ключа.

4. Щелкните нужную связь в списке **Выбранный элемент Отношение**.
5. Щелкните **Изменение свойств у существующего отношения** в сетке справа и нажмите кнопку с многоточием (...) справа от свойства.
6. В диалоговом окне **Таблицы и столбы** в раскрывающемся списке **Первичный ключ** выберите таблицу, которая будет находиться на стороне первичного ключа связи.
7. В сетке внизу выберите столбцы, составляющие первичный ключ таблицы. В соседней ячейке сетки справа от каждого столбца выберите соответствующий столбец внешнего ключа таблицы внешнего ключа.
8. Конструктор таблиц предложит имя для связи. Чтобы его изменить, отредактируйте содержимое текстового поля **Имя связи**.
9. Нажмите кнопку OK, чтобы создать связь.

Для ознакомления с проверочными ограничениями см. Конспект лекций п. 2.10.

- В диаграмме базы данных щелкните правой кнопкой таблицу, которая будет содержать ограничение и выберите пункт **Проверочные ограничения** из контекстного меню.

-или-

- Откройте таблицу, которая будет содержать ограничение в конструкторе таблиц, щелкните правой кнопкой мыши в конструкторе и выберите пункт **Проверочные ограничения** из контекстного меню.
- Нажмите кнопку **Добавить**.

Примечание. *Чтобы назвать ограничение по-другому, введите имя в поле Имя ограничения.*

- В поле **Выражение** в сетке введите SQL-выражения для проверочного ограничения. Например, чтобы ограничить записи в столбце state в таблице authors Нью-Йорком, введите: state = 'NY', а чтобы ограничить записи в столбце zip записями, со-

стоящими из 5 цифр, введите: zip LIKE '[0-9][0-9][0-9][0-9][0-9]'.

Примечание. *Убедитесь, что все нечисловые ограничения по значению заключены в одиночные кавычки ().*

- Разверните ***категорию*** конструктора таблиц, чтобы настроить момент выполнения проверочного ограничения.
- Чтобы проверить выполнение ограничения для данных, которые существовали до создания ограничения, отметьте флажок ***Проверка существующих данных при создании или повторном включении.*** Чтобы ограничение проверялось всякий раз, когда происходит добавление или обновление строки в этой таблице, отметьте флажок ***Принудительное использование для запросов INSERT и UPDATE.***

Проверка существующих данных при создании проверочного ограничения

- На диаграмме базы данных щелкните таблицу, содержащую ограничение, правой кнопкой мыши и в контекстном меню выберите **Проверочные ограничения.**

-или-

- Откройте таблицу, содержащую ограничение, в конструкторе таблиц, щелкните конструктор таблиц правой кнопкой мыши и в контекстном меню выберите **Проверочные ограничения**.
- Выберите ограничение в списке **Выбранные проверочные ограничения.**
- Щелкните **Проверка существующих данных при создании или повторном включении** и выберите **Да** в раскрывающемся списке.

3.4 Ввод данных посредством графического интерфейса

1. **Заполнение таблицы Types**. Сделайте щелчок правой кнопкой на таблице Types в Object Explorer и выберите **Открыть таб-**

лицу. Таблица откроется в окне редактора со строкой, подготовленной для ввода данных. Введите в таблицу Types две строки данных в порядке следования полей:

1 Мониторы
2 Принтеры

2. **Заполнение данными таблицы Suppliers** (Поставщики). Таким же образом заполним две строки в таблице Suppliers (Поставщики):

1, Tecnis, Вероника Кудрявцева, Менеджер по закупкам, ул. Большая Садовая, 12, Москва, 123456, Россия, (095) 325-2222, (095) 325-2222;

2, Uni, Дмитрий Сидоров, Координатор, Бостон 78934, Новый Орлеан, 70117, США, (100) 555-4822;

3. **Заполнение данными оставшихся таблиц** будет выполнено в следующей лабораторной работе.

4. Выполните очередное резервное копирование в соответствии с уже известным порядком действий (Лабораторная работа №1, п. 3.10).

4 Контрольные вопросы

1) Для чего используются диаграммы базы данных?
2) Как создать и сохранить новую диаграмму базы данных?
3) Сколько диаграмм можно создать для одной базы данных?
4) Укажите последовательность действий по включению в диаграмму новой таблицы?
5) Что такое "Ограничение внешнего ключа"? Укажите последовательность действий по созданию на диаграмме базы данных ограничения внешнего ключа.
6) Какие типы проверочных ограничений Вы знаете? Укажите последовательность действий по добавлению проверочного ограничения к таблице визуальными средствами конструктора базы данных.

ЛАБОРАТОРНАЯ РАБОТА № 3

СОЗДАНИЕ СЦЕНАРИЕВ В СРЕДЕ MANAGEMENT STUDIO

1 Цель работы

1) Научиться применению инструкции INSERT.
2) Научиться применению инструкции UPDATE.
3) Научиться применению инструкции DELETE.

2 Задание

1) Заполнить данными все таблицы созданной базы данных BOREI с использованием оператора INSERT и графического интерфейса Management Studio.
2) Изучить правила выполнения сценария посредством интерфейса Management Studio.
3) Выполнить примеры по изменению данных.
4) Выполнить примеры по удалению данных.

3 Порядок выполнения работы

3.1 Ввод данных посредством выполнения инструкции INSERT

1. Используя графический интерфейс **Обозревателя объектов**, последовательно откройте таблицы Types и Suppliers и удалите введённые ранее в них строки.
2. Откройте редактор запросов (если необходимо см. Конспект лекций п.п.4.1.1), нажав на панели инструментов Management Studio кнопку New Query.
3. ***Проверьте и, если необходимо, выберите из выпадающего списка на панели инструментов редактора запросов (см. номер 1 на рисунке 5.1 Конспекта лекций) базу данных, для которой будет выполнен запрос.***

4. Введите в окно редактора запросов и выполните следующий код:

```
SET IDENTITY_INSERT Types ON;
```

Выполнять запрос необходимо, нажав кнопку *или F5.*

***Примечание**. Данная строка необходима нам, если это свойство установлено для столбца первичного ключа, и мы собираемся вручную задать значение ключевого поля CodeType. Если предоставить серверу самостоятельно задать значение ключевого поля при добавлении новой строки, то эту часть запроса можно не выполнять.*

5. Теперь введите и выполните инструкцию, которая добавляет строку в таблицу Types:

```
INSERT INTO Types (CodeType, Category) VALUES (1, 'Мониторы')
```

***Примечание**. Помните, что строки символов, заключённые кавычки, являются чувствительными к регистру.*

6. Введите и выполните следующий код, добавляющий строку в таблицу Suppliers:

```
SET IDENTITY_INSERT Suppliers ON
```

Теперь введём код для создания строк:

```
INSERT INTO Suppliers (CodeSuppliers, Title, AddressTo, Post, Address, City, IIndex, Country, Telephone, Fax) VALUES (1, 'Tecnis', 'Вероника Кудрявцева', 'Менеджер по закупкам', 'ул. Большая Садовая, 12', 'Москва', '123456', 'Россия', '(095) 325-2222', '(095) 325-2222');

INSERT INTO Suppliers (CodeSuppliers, Title, AddressTo, Post, Address, City, IIndex, Country, Telephone) VALUES (2, 'Uni', 'Дмитрий Сидоров', 'Координатор', 'Бостон 78934', 'Новый Орлеан', '70117', 'США', '(100) 555-4822');
GO
```

Обратите внимание, что значение для IIndex является строковым, а не числовым.

7. Проверим правильность ввода наших данных, открыв таблицы в **Обозревателе объектов** путём выбора **Открыть таблицу** во всплывающем меню.
8. Для ввода остальных данных в таблицы Types, Suppliers, Clients, Deliveries, Employees, Goods, Orders и Ordered выполним следующие действия:
   - скопируем и сохраним в файле Insert.sql текст, приведённый в Приложении 1 настоящего руководства;
   - откроем сохранённый файл Insert.sql в Management Studio и выполним его (см. Конспект лекций п.п.4.1.1).
9. Просмотрим результат, открыв последовательно таблицы в редакторе Management Studio. При этом проверим правильность введенных данных, обратив внимание на то, нет ли полей, содержащих только значения NULL. Такие столбцы будут обнаружены в таблице Goods: Supplier, Category; в таблице Orders: Client, Employee, Delivery, а также в таблице Ordered: Goods и Price. Их нужно заполнить, обратившись к соответствующим таблицам и воспользовавшись имеющимися значениями: CodeClient, CodeEmployee, CodeGoods, что и будет выполнено в следующем подразделе 3.2.

3.2 Обновление (изменение) данных

1. Прежде всего, проанализируем и выполним нижеследующую инструкцию, в которой сначала выполняется инструкция SELECT (см. Лабораторную работу № 4), записанная в скобках после знака присвоения. Она выбирает из таблицы Suppliers значения столбца Title, у которых совпадают значения кодов в таблице Suppliers и в таблице Goods. Затем полученный результат заносится в столбец Supplier таблицы Goods.

UPDATE Goods SET Supplier = (SEI ECT Title FROM Suppliers WHERE Goods.CodeSuppliers=Suppliers.CodeSuppliers)

2. Откроем таблицу Goods и просмотрим результат. Столбец Supplier, содержавший ранее значения NULL, теперь заполнен соответствующими данными из таблицы Suppliers.
3. По аналогии с заполнением данными посредством выполнения сценария в п. 3.1 выполним следующие действия:
   - скопируем и сохраним в файле Updates.sql текст, приведённый в Приложении 2 настоящего руководства;
   - откроем сохранённый файл Updates.sql в редакторе запросов Management Studio и выполним его.
4. Проверим результаты, открыв соответствующие таблицы для просмотра.
5. Если предыдущие действия были выполнены успешно, то можно сделать очередную резервную копию базы данных backup2.bak (Лабораторная работа №1, п. 3.10).

3.3 Удаление данных

➡**Пример 1. Удаление строк из таблицы Ordered.** В этом упражнении вначале просмотрим заказы дешевле 1$. Затем соответствующие заказы будут удалены из таблицы Ordered, и мы убедимся, что это сделано.

1. Введём следующий оператор, чтобы просмотреть продажи по заказам:

SELECT * FROM Ordered WHERE Price < 1

Должно существовать три таких заказа.

2. Перед выполнения следующих действий сделайте копию базы данных. Введем следующий оператор DELETE:

DELETE FROM Ordered WHERE Price < 1

***Замечание**. Обратите внимание, что оператор DELETE не требует определения столбцов, поскольку он удаляет всю строку.*

3. Повторим первоначальный оператор SELECT. Он должен вернуть теперь пустой результат.
4. Для восстановления данных воспользуйтесь предыдущей копией, созданной перед выполнением оператора DELETE.
5. Выполним оператор SELECT снова, чтобы увидеть удалённые три строки.
6. После удачного выполнения действий по удалению и восстановлению записей уместно сделать очередную резервную копию базы данных backup4.bak

➧**Пример 2. Удаление записей на основе более точных условий.** Можно определить условие удаления более точно путём комбинирования ряда простых условий в одном. Например, введите следующий оператор, чтобы удалить записи, относящиеся к продаже товаров фирме "Соло", выполненным после 2 января 1994 г.:

```
DELETE FROM Orders WHERE DatePurpose>'02.01.1994' AND
Client='Соло'
```

Если попытаться выполнить этот оператор, то MS SQL Server вернёт ошибку, поскольку существует внешний ключ в таблице Ordered, который ссылается на таблицу Orders. Если эти строки будут удалены, то некоторые строки таблицы Ordered не будут больше иметь соответствующих строк в таблице Orders, что приведёт к нарушению ограничения внешнего ключа (нарушению ссылочной целостности).

***Примечание**. В операторах удаления можно использовать подзапросы, также как они использовались при обновлении данных.*

4 Контрольные вопросы

1) Укажите последовательность действий для визуального ввода данных в конкретную таблицу базы данных.

2) Как удалить строку данных из таблицы базы данных визуальными средствами?
3) Прокомментируйте смысл предложений в составе инструкции INSERT INTO.
4) Укажите последовательность подготовительных действий для ввода и выполнения инструкции языка Transact-SQL.
5) Дайте определение терминов ПАКЕТ и СЦЕНАРИЙ при выполнении инструкций языка Transact-SQL.
6) Расскажите о порядке выполнения сценария языка Transact-SQL на примере заполнения данными таблиц базы данных Borei.
7) Расскажите о порядке обновления (изменения) данных.
8) Расскажите о порядке удаления данных.

ЛАБОРАТОРНАЯ РАБОТА № 4

ОБЗОР ЯЗЫКА TRANSACT SQL

1 Цель работы

1) Научиться созданию запросов посредством конструктора запросов Management Studio.
2) Научиться правильному применению условий поиска данных в базе данных.

2 Задание

1) Изучить порядок открытия конструктора запросов в среде Management Studio.
2) Изучить интерфейс конструктора запросов.
3) Изучить правила визуального создания запросов посредством конструктора запросов и представлений.
4) Изучить и выполнить предложенные варианты условий поиска в составе инструкции SELECT.

3 Порядок выполнения работы

3.1 Создание и выполнение запроса к базе данных посредством конструктора запросов и представлений

Для создания нового запроса обычно выполняют нижеперечисленную последовательность шагов.

1. В **обозревателе объектов** сделайте щелчок на узле той базы данных, к которой Вы намерены создать запрос.
2. Сделайте щелчок на кнопке **Создать запрос** панели инструментов.
3. В позиции главного меню **Запрос** или во всплывающем меню окна открывшегося редактора выберите пункт **Создать**

запрос в редакторе. Откроется окно конструктора и окно переднего плана **Добавление таблицы**, содержащее перечень доступных таблиц.

4. В диалоговом окне **Добавление таблицы** выберите таблицы, к которым будет обращаться запрос, и для каждой из них нажмите кнопку **Добавить**.
5. Выбрав нужные таблицы, нажмите кнопку **Закрыть**. Изображения отобранных таблиц появятся в области диаграммы конструктора запросов.
6. Если позднее в запрос нужно будет добавить еще какие-нибудь таблицы, в меню конструктора запросов выберите пункт **Добавить таблицу**,

   или

   щелкните открытое место на панели **Диаграмма** правой кнопкой мыши и из контекстного меню выберите **Добавить таблицу**.

Примечание. *Если панели* ***Диаграмма, SQL, Критерий*** *или* ***Результаты*** *отсутствуют, в меню конструктора запросов выберите* ***Область*** *и щелкните панель, которую нужно открыть.*

7. На панели **Диаграмма** на изображениях таблиц поставьте флажки для каждого столбца, который должен войти в результат запроса. В результате этих действий в области **Критериев** автоматически будут заполнены указанные столбцы, а соответствующая инструкция SELECT также будет сгенерирована и появится в области SQL-кода.
8. Чтобы выполнить запрос, в меню конструктора запросов выберите **Выполнить SQL**.

При дальнейшем улучшении запроса можно изменить код SQL на панели **SQL** или выбрать такие параметры, как порядок сортировки или псевдонимы столбцов на панели **Критерии**.

Для открытия конструктора запросов и представлений для *существующего запроса*:

1. В обозревателе объектов разверните папку Queries.

2. Дважды щелкните мышью открываемый запрос. Области конструктора запросов и представлений открываются с учетом параметров, заданных в диалоговом окне

В качестве примера создадим запрос к таблицам Orders, Ordered для определения состава товаров, которые заказывала определённая фирма, например, E-Life. Таблица Orders содержит необходимую нам информацию о заказах, за исключением наименования товара, скидки, стоимости и количества, которая хранится в таблице Ordered. Из построенной ранее диаграммы видно, что таблица Ordered имеет ограничение внешнего ключа по отношению к таблице Orders. Для построения такого запроса с помощью конструктора запросов выполним следующую последовательность действий.

1. В обозревателе объектов делаем щелчок на узле **Таблицы** в составе дерева Borei.
2. На верхней панели инструментов делаем щелчок на кнопке **Создать запрос.**
3. На нижней панели инструментов открывшегося редактора запросов в выпадающем списке проверяем установку текущей базы – Borei.
4. В главном меню **Запрос** или во всплывающем меню окна редактора запросов выбираем **Создать запрос в редакторе.**
5. Выбираем таблицы *Ordered и Orders* в окне переднего плана, щёлкаем на кнопке **Добавить** и затем **Закрыть.** Выбранные таблицы появятся в области Диаграммы с отображением отношения между ними.
6. На панели критериев отбираем столбцы: *Orders.Employee, Orders.DateExecution. Ordered.Goods. Ordered.Discount, Ordered.Price. Ordered.Amount.*
7. Ставим галочки против них в столбце **Вывод.**
8. Добавляем поле Orders.Client и в столбец **Фильтр** заносим значение 'E-Life'. В столбец вывода для этого поля галочку можно не ставить, если мы не хотим отображать этот столбец.
9. Изучаем созданную инструкцию SELECT в области SQL.

10. Выполняем инструкцию щёлкнув на кнопке **Выполнить**.

Если все действия были правильными, то в области SQL-кода появится следующая инструкция:

SELECT ordered.Goods, ordered.Discount, ordered.Amount, ordered.Price, orders.Employee, orders.dateExecution **FROM** ordered **INNER JOIN** orders **ON** ordered.Codeorder = orders.Codeorder **WHERE** Orders.Client='E-Life',

а в области результатов будет выведен следующий результат:

Крылова Анна 1992-01-05 00:00:00.000 TEAC 52x CD-522E 0.0000 42 12

3.2 Практическое изучение приемов применения условий поиска

Перед выполнением последующих примеров следует ознакомиться с материалом, изложенным в п. 4.2 Конспекта лекций. Однако и здесь будет полезно кратко напомнить основные понятия, которые там рассматривались.

3.2.1 Оператор SELECT

Оператор SELECT является сердцем SQL, поскольку именно посредством него мы получаем обратно информацию, которую сохраняли в базе данных. Нет смысла создавать и заполнять структуры данных, если мы не сможем получить их обратно в приемлемой форме. Мы уже встречались с некоторыми простыми формами оператора SELECT в предыдущей лаб. работе. В данной части руководства Вы расширите практику в его использовании.

Рассмотрим укрупненный синтаксис оператора SELECT:

```
SELECT [DISTINCT] columns
FROM tables
WHERE < search_conditions>
[GROUP BY column [HAVING < search_condition>]]
[ORDER BY < order_list> [ ASC | DESC ]]
```

Оператор SELECT содержит семь главных ключевых слов. Ключевое слово и связанную с ним информацию называют ***предложением***. Эти предложения представлены в нижеследующей таблице 4.1.

Таблица 4.1 – Структура оператора SELECT

Предложение	Описание
SELECT columns	Список возвращаемых столбцов
DISTINCT	Опциональное ключевое слово, исключающее повторяющиеся строки
FROM tables	Указывает таблицы, из которых должны выбираться значения
WHERE <search_conditions>	Определяет условия отбора подмножества строк из множества всех доступных строк
GROUP BY column	Группирует возвращаемые строки на основе значений указанного столбца
HAVING <search_conditions>	Используется совместно с предложением GROUP BY и определяет условия отбора групп
ORDER BY <order_list>	Упорядочивает результирующий набор строк, возвращаемых оператором SELECT, на основе указанных столбцов

3.2.2 Условия поиска

Текст, следующий за ключевым словом WHERE, называется ***условиями поиска***, поскольку оператор SELECT ищет строки, удовлетворяющие этому условию поиска. Условие поиска состоит из имени столбца (например "Surname"), оператора (например "=") и значения (например "Белова"). Таким образом, предложение WHERE имеет следующую общую форму:

WHERE column_name operator value

В общем случае column_name есть имя столбца запрашиваемой таблицы, operator — это оператор сравнения и value — это значение или диапазон значений, с которым сравнивается значение столбца. Условия поиска, в общем случае, используют операторы, приведённые в таблице 4.2.

Таблица 4.2 – Классификация операторов в SQL Server

Операторы	Описание
Операторы сравнения	Используются для сравнения значения столбца со значением условия поиска. Примерами являются: <, >, <=, >=, =, != или < >, !< (не меньше), !> (не больше).
Арифметические операторы	Используются для вычисления и оценки значений в условии поиска. Примерами служат: +, -, *, /, % (остаток от деления).
Логические операторы	Проверяют истину некоторого условия. Это операторы: NOT, AND, OR, а также ALL, ANY или SOME, BETWEEN, IN, LIKE и EXISTS.

Значения в условии поиска могут быть литеральными (константы) или вычисляемыми, а также значениями, возвращаемыми подзапросами. Описание типов значений приведено в таблице 4.3.

Таблица 4.3 – Типы значений в условии поиска

Типы значений	Описание
Литеральные значения	Числа и символьные строки, являющиеся образцами для сравнения, например число 1138 или строка ‘Новиков’.
Вычисляемые значения	Функции или арифметические выражения, например Price * 2 или Surname || Name.

Окончание таблицы 4.3

Подзапросы (**подзапрос** - это вложенный в запрос оператор SELECT)	Вложенный оператор SELECT возвращает одно или более значений, которые используются для сравнения со значениями определённого столбца.

Рекомендуется символьные значения заключать в ***одинарные*** кавычки, поскольку в некоторых случаях двойные кавычки являются недопустимыми. Числовые константы записываются без кавычек.

Замечание. *Строковые значения являются чувствительными к регистру.*

Когда строка предварительного результата запроса подвергается проверке на соответствие условию, то результатом может быть одно из трёх значений:

- True – строка соответствует условию предложения WHERE;
- False – строка не соответствует условию предложения WHERE;
- Unknown -- некоторое поле в предложении WHERE содержит неопределённое значение NULL, которое не может быть оценено.

3.2.3 Примеры выполнения поиска

➡**Пример 1**. Выбрать строки со значением "Белова" в столбце Surname таблицы Employees. В результате ваших действий конструктор запросов должен сгенерировать следующую инструкцию:

```
SELECT Surname, Name, Post, HomeTelephone
FROM Employees
WHERE Surname='Белова'
```

В результате выполнения запрос должен вернуть одну строку:

Белова Мария Представитель (017) 555-9857

➡**Пример 2.** Выбрать строки, у которых фамилия (Surname) следует в алфавитном порядке за фамилией Белова. В результате ваших действий конструктор запросов должен сгенерировать следующую инструкцию:

```
SELECT Surname, Name, Post, HomeTelephone
FROM Employees
WHERE Surname > 'Белова'
```

В результате выполнения запрос должен вернуть одну строку:

Крылова Анна Внутренний координатор (017) 555-1189

➡**Пример 3. Поиск с отрицанием NOT.**

Любое логическое выражение можно превратить в его отрицание, поставив перед ним оператор NOT.

Например, выберите категории (типы) товаров за исключением CD-ROM-мов. В данном случае запрос будет выглядеть следующим образом:

```
SELECT CATEGORY FROM TYPES
WHERE NOT CATEGORY = 'CD-ROM'
```

В результате должен получиться список из 15 категорий.

Однако есть и другие способы получить тот же самый результат, поставив вместо NOT следующие операторы:

```
SELECT Category FROM Types WHERE CATEGORY != 'Cd-Rom'
SELECT Category FROM Types WHERE CATEGORY <> 'Cd-Rom'
```

Применение сравнения LIKE

Кроме сравнения непосредственно со значениями, условия поиска могут содержать специальный шаблон. Если проверяемые данные удовлетворяют данному шаблону, то строка включается в результат.

Сравнение **LIKE** является чувствительным к регистру и использует символы-заменители. Символ (%) соответствует в шаблоне любому набору символов. Символ (_) соответствует одному произвольному символу.

Таблица 4.4 содержит примеры некоторых общих шаблонов.

Таблица 4.4 -- Примеры шаблонов

Предложение WHERE	**Соответствует**
SURNAME LIKE '%a%'	Значение поля SURNAME содержит по крайней мере один символ “a”.
SURNAME 'Кр%'	Значение поля SURNAME начинается символами “Кр”.
CONTAINS (SURNAME, 'i')	Значение поля SURNAME содержит по крайней мере одну букву “i” или “I”.
SURNAME BETWEEN 'A' AND 'Н'	Значение поля SURNAME начинается с любой буквы между “А” и “Н” включительно.

➧**Пример 4.** Создать запрос для поиска служащих, чьи фамилии оканчиваются на “ова”. Вы должны будете получить следующую инструкцию:

```
SELECT SURNAME,NAME FROM EMPLOYEES
WHERE SURNAME LIKE '%ова'
```

В результате должен быть получен следующий результат:

```
SURNAME        NAME
==========================
Белова         Мария
Крылова        Анна
```

➧**Пример 5.** Создать запрос для поиска служащих, чьи фамилии начинаются с “К”, после которого следуют точно два символа и затем символ “л”. Остальные символы безразличны. Вы должны будете получить следующую инструкцию:

```
SELECT SURNAME, NAME FROM EMPLOYEES
WHERE SURNAME LIKE 'К__л%'
```

Результатом должно быть:

Крылова Анна.

Применение оператора CONTAINS

Оператор **CONTAINS** проверяет наличие заданной сроки символов в любом месте искомой строки. Этот оператор не является чувствительным к регистру и не поддерживает неопределённых символов, если записи в таблице написаны на латинском языке.

Примечание. *Для того, чтобы этот оператор отработал, необходимо, чтобы на сервере была включена служба полнотекстового поиска (full-text search service) и таблица была проиндексирована для него.*

➡**Пример 6.** Создать запрос на поиск клиентов, чьи названия содержат буквы "i" и "I" в любом месте. Созданная инструкция должна принять следующий вид:

```
SELECT TITLE FROM Clients WHERE CONTAINS (TITLE,'I')
```

Должен быть получен следующий результат:

```
TITLE
==========
 E-Life
IMC Computers
Comtris
Net Line
Oki
```

Теперь выполним тот же самый оператор, заменив букву "I" на "i". Мы должны будем получить тот же самый результат.

```
SELECT TITLE FROM Clients WHERE CONTAINS (TITLE,'i')
```

Этот оператор является чувствительным к регистру, если записи написаны "кириллицей".

```
SELECT SURNAME, NAME FROM EMPLOYEES
WHERE CONTAINS (NAME,'a')
```

Результатом будет:

SURNAME	NAME
Белова	*Мария*

Теперь выполним тот же самый оператор, заменив букву "а" на "А":

```
SELECT SURNAME, NAME FROM EMPLOYEES
WHERE CONTAINS (NAME,'A')
```

Получим:

SURNAME	NAME
Крылова	*Анна*

Проверка на неопределённое значение (оператор IS NULL)

Другой тип сравнения проверяет на наличие, или отсутствие какого-либо значения. Для этой цели используется оператор **IS NULL**. Для проверки присутствия некоторого значения используют оператор **IS NOT NULL**.

➧**Пример** 7. Создать запрос, который возвращает названия фирм, не имеющих факса:

```
SELECT TITLE, FAX FROM CLIENTS WHERE FAX IS NULL
```

Запрос должен вернуть названия четырёх фирм: *МАП Инфо, Дамодара-Сервис, IMC Computers, Stop.*

Выполним теперь оператор с использованием IS NOT NULL:

```
SELECT Title, Fax FROM CLIENTS WHERE FAX IS NOT NULL
```

Результат должен будет содержать следующие строки: SELECT Title, Fax FROM CLIENTS WHERE **FAX IS NOT NULL**

TITLE	FAX
Омикс	0921-12 34 67
Белкантон	(5) 555-7293
E-Life	7675-3426
Верса	089-0877451
Comtris	(2) 283-3397
Net Line	(21) 555-8765
NTTs	(5) 555-1948
ИнтеллектС	2967 3333
ZS	(9) 331-7256
Медиа-софт	035-640231
Эликон-М	(907) 555-2880
Соло	0522-556722
Olymp	(91) 745 6210
Oki	(11) 555-2168

Сравнение с диапазоном и списком значений

В предыдущей секции были рассмотрены операторы, которые выполняют сравнение с единственным значением, в то время как операторы **BETWEEN** и **IN** сравнивают искомое значение со множеством заданных значений. Оператор **BETWEEN** проверяет попадание искомого значения в заданный диапазон.

➧**Пример 8. Применение оператора BETWEEN.** Создать запрос, который возвращает все фамилии, начинающиеся с букв, расположенных между "А" и "К". Заметим, что запрос не включает фамилии, начинающиеся с последней буквы диапазона - "К". Это происходит потому, что оператору BETWEEN удовлетворяют значения меньшие или равные конечному значению диапазона. Поэтому фамилия, начинающаяся с "К" и содержащая ещё другие буквы, будет больше, чем "К". Инструкция должна принять вид:

```
SELECT SURNAME, NAME FROM EMPLOYEES
WHERE SURNAME BETWEEN 'A' AND 'K'
```

Результирующий набор должен будет содержать:

SURNAME	NAME
Белова	*Мария*

➡**Пример 9. Применение оператора BETWEEN.** Создать запрос, *который возвращает наименования товаров, цена которых* лежит в пределах между 14 и 32 включительно.

```
SELECT CATEGORY, SUPPLIER, PRICE FROM GOODS
WHERE PRICE BETWEEN 14 AND 32
ORDER BY PRICE
```

Результат должен состоять из 11 строк, включающих нижнюю и верхнюю границы диапазона.

CATEGORY	SUPPLIER	PRICE
Корпуса и блоки питания	SV-Trading	14
Модули памяти	Uni	15
Устройства ввода и указания	Гвин-Медиа	15
Модемы	Iven	18
Модули памяти	ПК Сервис	19
Сетевое оборудование	DAAS	22
CD-ROM	B.S.T.Group	23
CD-ROM	Конструктив	25
Видеокарты	Stepfor	28
Сетевое оборудование	BelSoft	32
Сетевое оборудование	Гвин-Медиа	32

Поиск с использованием оператора IN

Оператор **IN** сравнивает искомые значения с одним из значений заданного списка. Значения списка должны отделяться друг от друга запятыми. Можно использовать оператор NOT для поиска значений, несовпадающих ни с одним из значений заданного списка.

➡**Пример 10.** Создать запрос, который возвращает названия товаров, имеющих следующие цены: 15, 19, 32.

```
SELECT CATEGORY, SUPPLIER, PRICE FROM GOODS
WHERE PRICE IN (15,19,32)
ORDER BY PRICE, CATEGORY
```

Должен быть получен следующий результирующий набор строк:

CATEGORY	SUPPLIER	PRICE
Модули памяти	Uni	15
Устройства ввода и указания	Гвин-Медиа	15
Модули памяти	ПК Сервис	19
Сетевое оборудование	BelSoft	32
Сетевое оборудование	Гвин-Медиа	32

Применение логических операторов AND и OR

Несколько условий поиска можно объединить в одном предложении WHERE посредством использования логических операторов **AND** и **OR**.

Когда два условия связываются оператором **AND**, то имеется в виду, что оба условия должны быть истинными для искомой строки.

➡**Пример 11. Посредством использования области критериев конструктора запросов** создать запрос, который возвращает служащих, работающих на должности представителя и принятых на работу до 1 января 1994 г. Должна быть сформирована следующая инструкция:

```
SELECT Surname, Name, Post, DateHiring, HomeTelephone
FROM Employees
WHERE Post='Представитель' AND DateHiring < '1.01.1994'
```

В результате оператор должен вернуть одну строку:

Белова Мария Представитель 1992-01-05 00:00:00.000 (017) 555-9857

➡**Пример 12. Посредством использования области критериев конструктора запросов** создать запрос, который возвращает клиентов из Германии или Италии. Для поиска строк, удовлетворяющих хотя бы одному условию среди нескольких заданных, используют логический оператор **OR**. Должа быть получена следующая инструкция:

```
SELECT Title, Address, City, Country
FROM Clients
WHERE (Country='Германия' OR Country='Италия')
```

Результат запроса должен содержать четыре строки:

Верса	*Берлинская пл., 43*	*Мюнхен*	*Германия*
Медиа-софт	*ул Людовика, 22*	*Бергамо*	*Италия*
Stop	*Тачерстрасс, 10*	*Кюневальд*	*Германия*
Соло	*ул. Провинциальная, 124*	*Реджио-Эмилио*	*Италия*

Управление порядком вычисления условий

Когда вводятся сложные условия поиска, следует быть уверенным в правильности порядка выполнения этих условий. Предположим, что мы хотим выбрать служащих, принятых на работу до 1993 или после 1993 и родившихся до 1960 года.

➡**Пример 13. Неудачная попытка выполнения составного условия.** Создайте и выполните следующий оператор:

```
SELECT Surname, Name, Post, DateBirth, DateHiring
FROM Employees
WHERE DateHiring < '01.01.1993' OR DateHiring > '12. 31.1993'
      AND DateBirth < '01.01.1960'
```

Можно увидеть, что в результате присутствуют только принятые на работу служащие моложе 1960 года:

Белова	*Мария*	*Представитель*	*08.12.1968*	*01.05.1992*
Крылова	*Анна*	*Внутренний координатор*	*09.01.1958*	*05.03.1994*

Этот запрос возвращает неожиданный результат, потому что логический оператор **AND** имеет более высокий приоритет, чем опе-

ратор **OR**. Это значит, что выражение с оператором AND выполняется раньше выражения с оператором OR.

➧**Пример 14. Успешная попытка выполнения составного условия.** Для изменения нормального приоритета выполнения операций используют скобки. В нижеследующем упражнении скобки размещаются вокруг двух дат приема на работу, чтобы они проверялись оператором AND, как единое целое. Выполним предыдущий оператор, но с правильной расстановкой скобок:

```
SELECT Surname, Name, Post, DateBirth, DateHiring
FROM Employees
WHERE (DateHiring< '01.01.1993' OR DateHiring> '12.31.1993')
      AND DateBirth < '01.01.1960'
```

Теперь мы получили желаемый результат:

Крылова Анна Внутренний координатор 09.01.1958 05.03.1994

***Примечание**. Учёт приоритета выполнения операций важен не только при использовании операторов **AND** и **OR**, все операции имеют определённый приоритет, который управляет порядком их выполнения. Этот порядок определяется в описании любого языка программирования и, в частности, языка SQL. В случае неуверенности в правильности выполнения операций следует применять скобки.*

4 Контрольные вопросы

1) Что такое условие поиска и где оно применяется?
2) Объясните правила применения шаблонов в условиях поиска.
3) Как выполняется проверка на неопределённое значение?
4) Как формулируется условие поиска для проверки на диапазон допустимых значений?
5) Как формулируется условие поиска для проверки на перечень конкретных значений?
6) Объясните правила управления порядком вычисления логических условий.

ЛАБОРАТОРНАЯ РАБОТА № 5

ИСПОЛЬЗОВАНИЕ ПОДЗАПРОСОВ В ЯЗЫКЕ SQL

1 Цель работы

1) Научиться использовать подзапросы в составе инструкций Transact-SQL.
2) Научиться применять кванторы в условиях поиска данных в базе данных.
3) Научиться применять в запросах агрегатные функции.

2 Задание

1) Изучить правила и условия применения подзапросов в составе инструкций Transact-SQL и выполнить предложенные примеры.
2) Изучить правила и условия применения кванторов и выполнить предложенные примеры.
3) Изучить правила и условия применения в запросах агрегатных функций и выполнить предложенные примеры.

3 Порядок выполнения работы

3.1 Использование подзапросов

Подзапросы являются специальным случаем предложения WHERE, но в силу важности этого инструмента они заслуживают отдельного обсуждения.

Вспомним, что в предложении WHERE записывается имя столбца, оператор сравнения и значение, с которым осуществляется сравнение. SQL сервер сравнивает содержимое столбца с заданным значением посредством указанного оператора. Вместо значения, с которым производится сравнение в предложении WHERE, можно

использовать оператор SELECT. Внутреннее предложение SELECT называют ***подзапросом***. SQL сервер выполняет подзапрос и использует его результат в качестве значения для сравнения в предложении WHERE.

Предположим, например, что мы хотим выбрать список всех товаров фирмы, которая поставляет принтеры Lexmark. Без использования подзапроса вначале нужно определить фирму:

```
SELECT Supplier
FROM Goods
WHERE Mark='Lexmark Z35'
```

Этот запрос вернёт "SV-Trading". Используя этот результат можно сформировать второй запрос, чтобы выбрать все товары фирмы "SV-Trading":

```
SELECT Mark
FROM Goods
WHERE Supplier='SV-Trading';
```

Использование подзапроса позволяет объединить оба запроса в один оператор.

➡**Пример 1.** *Выполнить подзапрос, возвращающий единичное значение.*

```
SELECT Mark
FROM Goods
WHERE Supplier = (SELECT Supplier FROM Goods
WHERE Mark='Lexmark Z35')
```

В этом случае подзапрос возвращает единственное значение "SV-Trading", в качестве значения, с которым будет осуществляться сравнение в предложении WHERE. Подзапрос ***должен вернуть единственное значение***, поскольку в предложении WHERE осуществляется проверка на знак равенства ("="). В противном случае запрос сгенерирует ошибку. Результирующий набор для обсуждаемого оператора будет состоять из трех наименований товаров.

Подзапросы с множественным результатом

Если подзапрос возвращает более одного значения, то содержащее его предложение WHERE должно использовать оператор, осуществляющий сравнение с несколькими значениями. Оператор IN является именно таким оператором.

➧**Пример 2.** Выполнить следующий пример, в котором выбирается название товара и фирмы, заказавшей какой-либо товар фирмы «SV-Trading». В этом примере используется подзапрос, который возвращает все товары фирмы «SV-Trading». Главный запрос, в свою очередь, выбирает клиента и проверяет, является ли он клиентом «SV-Trading».

```
SELECT Client, Goods
FROM Orders, Ordered
WHERE Goods IN (SELECT Mark FROM Goods
        WHERE Supplier='SV-Trading')
        AND Orders.CodeOrder=Ordered.CodeOrder
```

Результирующий набор должен выглядеть следующим образом:

Client	*Goods*
Соло	*Lexmark Z35*

3.2 Применение кванторов в подзапросах

Нижеследующая таблица 5.1 обобщает кванторы, которые сравнивают выражение в левой части условия с результатом выполнения подзапроса, содержащегося в правой части.

Таблица 5.1 – Классификация кванторов в SQL Server

Оператор	Действие
ALL	Принимает значение "истина", если сравнение возвращает истину для всех результирующих значений подзапроса.

Окончание таблицы 5.1

ANY или SOME	Принимает значение "истина", если сравнение возвращает истину хотя бы для одного результирующего значения подзапроса.
EXISTS	Определяет, существует ли хотя бы одно значение в результате подзапроса.

➡**Пример 1. Применение ALL.** Предположим, что мы хотим найти товары, которые дороже всех товаров фирмы «CD-Life». Введём следующий запрос:

```
SELECT Supplier, Mark, Price
FROM Goods
WHERE Price > ALL(SELECT Price FROM Goods
                  WHERE Supplier='CD-Life')
```

Результат должен выглядеть следующим образом:

Supplier	Mark	Price
BelSoft	Canon LBP810	185
DAAS	Samsung 550B	150
Q-Senter	Intel P4 1700MHz Box	137
Ситипринт	Samtron 76E	180

Этот пример использует квантор **ALL**. Значение столбца Price в каждой строке таблицы проверяется для каждого значения подзапроса. Если цена больше всех значений подзапроса, то строка помещается в результирующий набор.

➡**Пример 2. Применение ANY, EXISTS.** Вместо проверки условия для всех значений подзапроса, **запрос** можно переписать таким образом, чтобы условие выполнялось, по крайней мере, для одного значения:

```
SELECT Supplier, Mark, Price
```

```
FROM Goods
WHERE Price > ANY(SELECT Price FROM Goods
WHERE Supplier='CD-Life')
```

Этот оператор должен вернуть 14 строк, в которых цена больше хотя бы одного значения подзапроса.

Ключевое слово **ANY** является синонимом **SOME**. Они являются взаимозаменяемыми.

Ещё одним квантором является **EXISTS**. Он осуществляет проверку на существование хотя бы одной строки, удовлетворяющей условиям подзапроса, и возвращает соответственно TRUE или FALSE, даже если строки содержат значения NULL.

3.3 Применение агрегатных функций

SQL содержит агрегатные функции, которые вычисляют единичное значение на основе группы значений. Группой значений могут являться данные определённого столбца для заданного набора строк. Агрегатная функция может использоваться в предложении SELECT или где угодно вместо значения в операторе SELECT.

Нижеследующая таблица 5.2 перечисляет некоторые агрегатные функции, поддерживаемые SQL сервером.

Таблица 5.2 – Агрегатные функции

Функция	Действие
AVG([ALL | DISTINCT] value)	Возвращает среднее значение поля для группы строк.
COUNT ([ALL | DISTINCT] value | *)	Осуществляет подсчет количества строк, при этом в полях, входящих в *value*, игнорируется значение NULL.
MIN([ALL | DISTINCT] value)	Возвращает минимальное значение в столбце для группы строк.
MAX([ALL | DISTINCT] value)	Возвращает максимальное значение в столбце для группы строк.

Окончание таблицы 5.2

SUM([ALL | DISTINCT] value)	Возвращает сумму всех значений *value* для группы строк.

➧**Пример 1**. Предположим, что мы хотим узнать, сколько имеется различных поставщиков в таблице Goods. Введём следующий оператор:

```
SELECT COUNT(Supplier) FROM Goods
```

В качестве результата будет выведено одно значение – 45.

Однако это не тот результат, который нам нужен, поскольку он включает повторяющиеся значения. Чтобы подсчитать только уникальные названия поставщиков следует использовать ключевое слово DISTINCT:

```
SELECT COUNT (DISTINCT Supplier) FROM Goods
```

Будет получен другой правильный результат.

```
COUNT
======
19
```

Замечание. *Полученное число поставщиков равно количеству записей в таблице Suppliers.*

➧**Пример 2**. Вычислить среднюю стоимость товаров в Goods:

```
SELECT AVG(Price) FROM Goods
```

Результат будет иметь следующий вид:

```
AVG
==========
45,568000
```

➧**Пример 3**. В одном операторе SELECT можно применять несколько агрегатных функций. Введём следующий оператор, который подсчитывает число служащих и сотрудника с наибольшим стажем:

```
SELECT COUNT(Surname), MIN(DateHiring)
FROM Employees
```

Результат будет иметь вид:

```
COUNT       MIN
==========  ==========
2           1992-01-05
```

Замечание. *Если некоторое значение, вовлечённое в подсчёт агрегатной функцией, есть NULL или является неизвестным, то данная строка игнорируется целиком, чтобы избежать фатальной ошибки. Например, если вычисляется среднее значение для пятидесяти строк, десять из которых содержат значение NULL, то в действительности это значение будет являться средним значением для сорока строк.*

➧**Пример 3**. Чтобы увидеть, как агрегатная функция игнорирует строки, содержащие значения NULL выполним следующий тест.

```
SELECT FAX FROM SUPPLIERS
```

Полученный результат состоит из 19 строк, 11 из которых равны NULL. Подсчитаем число значений в столбце Fax:

```
SELECT COUNT(Fax) FROM Suppliers
```

Результат будет содержать число восемь, а не 19.

3.4 Группирование и упорядочение результата запроса

3.4.1 Применение предложения ORDER BY

Строки таблиц базы данных не хранятся в каком-либо определённом порядке. После выполнения запроса можно обнаружить, что порядок следования строк нас не устраивает. Предложение **ORDER BY** позволяет нам указать желаемый порядок следования строк в результате запроса. Также можно применить предложение GROUP BY, чтобы сгруппировать результаты агрегатных функций.

Для сортировки можно использовать один или более столбцов

посредством указания имён или порядковых номеров. Предложение **ORDER BY** имеет следующий синтаксис:

ORDER BY [col_name | int] [ASC[ENDING] | DESC[ENDING]] [, ...]

Обратим внимание на то, что можно определить несколько столбцов, на основании которых будет осуществляться сортировка, а также на то, что ссылаться на столбцы можно не только по их именам, но и по порядковым номерам их следования в предложении SELECT. По умолчанию SQL сервер использует возрастающий порядок (ASCENDING), однако мы можем определить и убывающий (DESCENDING) порядок.

➡**Пример 1.** Выполним следующий запрос:

```
SELECT Mark, Price
FROM Goods
WHERE Price > 70
```

Здесь не определен какой-либо порядок вывода результирующих строк, и результат будет иметь следующий вид:

Mark	Price
HDD Seagate Baracuda 4 40Gb	87
HDD IBM 60Gb	79
Canon LBP810	185
Samsung 550B	150
Epson EPL520	90
Intel P4 1700MHz Box	137
Samtron 76E	180
Asus A7S333	86
TV-tuner Aver MediaTV Studio	75

➡**Пример 2.** Выполним тот же самый запрос, но с сортировкой по столбцу Mark:

```
SELECT Mark, Price
FROM Goods
```

```
WHERE Price>70
ORDER BY Mark
```

Обратим внимание, что, результирующий набор отсортирован по столбцу Mark в алфавитном порядке по возрастанию.

➧**Пример 3.** Упорядочим теперь результирующий набор по столбцу Price таблицы Goods:

```
SELECT Mark, Price
FROM Goods
WHERE Price>70
ORDER BY Price
```

➧**Пример 4.** Выполним предыдущий запрос, но изменим порядок сортировки на убывающий:

```
SELECT Mark, Price
FROM Goods
WHERE Price>70
ORDER BY Price DESC
```

➧**Пример 5.** Чтобы увидеть эффект от сортировки по нескольким столбцам, выполним очередной запрос:

```
SELECT Surname, Name, HomeTelephone FROM Employees
ORDER BY Surname DESC, Name
```

Обратим внимание, что две строки фамилий упорядочены в убывающем порядке, а имена — в возрастающем по умолчанию.

3.4.2 Применение предложения GROUP BY

Когда мы применяем запрос (оператор SELECT), содержащий как агрегатные функции (AVG, COUNT, MIN, MAX, или SUM), так и обычные столбцы, необходимо использовать предложение GROUP BY для группирования результирующего набора по каждому неагрегатному столбцу. При этом применяются три правила:

- каждый не агрегированный столбец результирующего набора должен появиться в предложении GROUP BY;
- предложение GROUP BY может ссылаться только на столбцы, которые упомянуты в предложении SELECT;
- каждое предложение SELECT в запросе может иметь только одно предложение GROUP BY.

Группы определяются как подмножества строк, соответствующих данному значению столбца, определённому в предложении GROUP BY.

➧**Пример. Группирование результирующего набора, содержащего агрегатные функции.** Выполним следующий запрос, чтобы определить количество сотрудников по каждой стране:

```
SELECT COUNT(CodeOrder), Client FROM Orders
GROUP BY Client
```

Результат должен иметь следующий вид:

COUNT	Client
1	E-Life
1	IMC Computers
1	Net Line
1	Olymp
1	Белкантон
2	Верса
1	ИнтеллектС
2	Соло
1	Эликон-М

3.4.3 Применение предложения HAVING

Также как предложение WHERE уменьшает количество строк, возвращаемых оператором SELECT, предложение HAVING может уменьшить число строк, возвращаемых предложением GROUP BY. Подобно предложению WHERE предложение HAVING содержит

условия поиска. Однако эти условия обычно применяются к агрегатным функциям, содержащимся в предложении SELECT.

➧**Пример 1. Управление запросом посредством GROUP BY и HAVING.** Применим следующий запрос, чтобы получить список клиентов, заказавших товаров на общую сумму свыше 100, и упорядочим результат по клиентам:

```
SELECT Client, SUM(Price) FROM Orders, Ordered
WHERE Orders.CodeOrder=Ordered.CodeOrder
GROUP BY Client
HAVING SUM(Price) > 100
ORDER BY Client
```

Результат должен иметь следующий вид:

```
Client      SUM
==========
Olymp       119
Соло        113
```

➧**Пример 2. Сортировка результирующего набора по результату агрегатной функции.** Но что, если в предшествующем примере мы захотели бы упорядочить результат не по клиентам, а по общей сумме заказов? Ведь в предложении ORDER BY должна быть ссылка на имя столбца из предложения SELECT, в то время как в нашем примере столбец, содержащий общую сумму заказов, представлен именем агрегатной функции. В таком случае мы должны воспользоваться второй альтернативой синтаксиса предложения ORDER BY (ORDER BY [col_name | int]) — использовать порядковый номер столбца в предложении SELECT.

Вернёмся к последнему запросу и изменим предложение ORDER BY в соответствии со следующим примером:

```
SELECT Client, SUM(Price) FROM Orders, Ordered
WHERE Orders.CodeOrder=Ordered.CodeOrder
GROUP BY Client
HAVING SUM(Price) > 100
ORDER BY 2
```

Результирующий набор должен быть отсортирован по второму столбцу в возрастающем порядке:

Client	SUM
Соло	113
Olymp	119

4 Контрольные вопросы

1) Объясните, что такое подзапрос и чем он отличается от запроса.
2) В каком предложении инструкции SELECT можно использовать подзапрос?
3) Можно ли применять подзапрос в Инструкции Transact-SQL INSERT?
4) Можно ли применять подзапрос в Инструкции Transact-SQL UPDATE?
5) Можно ли применять подзапрос в Инструкции Transact-SQL DELETE?
6) Для чего служит инструкция ORDER BY и как её определить визуальными средствами?
7) Для чего служит инструкция GROUP BY и как её определить визуальными средствами?
8) Перечислите состав агрегатных функций и укажите порядок их применения.

ЛАБОРАТОРНАЯ РАБОТА № 6

СОЗДАНИЕ ПРЕДСТАВЛЕНИЙ

1 Цель работы

1) Научиться создавать и использовать представления.

2 Задание

1) Изучить правила создания и применения представлений и выполнить предложенные примеры.

3 Порядок выполнения работы

3.1 Создание представлений

Представление (View) представляет собой виртуальную таблицу, содержащую выбранные строки и столбцы из одной или нескольких таблиц или других представлений.

MS SQL Server хранит только определения представлений.

Представление часто работает как секретное устройство, поскольку можно предоставить пользователям разрешение работать с представлением, а не с оригинальными таблицами. Таким образом, пользователи могут работать только с данными, имеющимися в представлении, в то время как остальные данные остаются недоступными.

Представления обычно используют для сохранения часто применяемых запросов или наборов запросов к базе данных. Выборка из представления осуществляется так же, как из таблиц, хотя другие операции имеют ряд ограничений.

В следующем упражнении будет использована инструкция **CREATE VIEW** языка Transact-SQL для создания списка поставщиков со всеми поставляемыми ими товарами путём выборки Title,

Address, Telephone из таблицы Suppliers, а также Category, Mark, Price, InWarehouse, Expected, MinimalStock, DeliveriesStopped из таблицы Goods.

Так как представление сохраняется в базе данных, а запрос — нет, процесс создания нового представления отличается от процесса создания запроса.

Для создания представления:

1. в обозревателе объектов щелкните правой кнопкой мыши узел **Представления** и в контекстном меню выберите пункт **Добавить новое представление**.
2. Продолжите конструировать представление так же, как запрос SELECT.

Примечание. *Однако в отличие от конструирования запроса SELECT, для представлений имеются некоторые ограничения. Дополнительные сведения см. в разделе Представления (п. 4.6. Конспекта лекций).*

Открыть представление можно с помощью команды Modify View или Open View.

Открытие результатов представления в области «Результаты» конструктора представлений:

1. в обозревателе объектов щелкните правой кнопкой мыши представление и выберите **Изменить представление**.
2. Откроется окно конструктора запросов и представлений, область «Результаты» которого отображает данные, содержащиеся в представлении. Для отображения других областей в меню Конструктора запросов выберите Область, а затем щелкните область, которую хотите открыть.

Открытие определения представления:

в обозревателе объектов щелкните правой кнопкой мыши представление и выберите **Проект (Изменить** в версии с пакетом обновления 1 или более ранней версии).

Примечание. *По умолчанию, конструктор запросов и представлений открывает все свои области («SQL», «Критерии», «Диаграмма» и «Результаты»), но эту настройку можно изменить в*

диалоговом окне ***Параметры****. Для открытия этих областей, если они закрыты, в меню* ***Конструктора запросов и представлений*** *укажите* ***Область*** *и выберите область, которую хотите открыть.*

Сохранение представления. При сохранении представления изменяется определение представления на сервере. В компоненте Database Engine используется новое определение представления при сохранении представления. Для сохранения представления:

1. откройте определение представления в конструкторе запросов и представлений и измените его.
2. В меню **Файл** выберите **Сохранить** view_name, где view_name — имя открытого представления.

Переименование представления. Изменить имя представления невозможно. Вместо этого необходимо создать новое представление с другим именем и скопировать в него определение старого представления. Это можно сделать посредством следующих шагов:

1. создайте новое представление с именем по вашему усмотрению.
2. Откройте старое представление в конструкторе запросов и представлений.
3. Скопируйте весь текст на панели SQL.
4. Вернитесь к новому представлению и вставьте текст.

Внимание! *При переименовании представления фрагменты кода и приложения, использующие это представление, могут привести к сбою. Это относится к запросам, представлениям, хранимым процедурам, пользовательским функциям и клиентским приложениям. Учтите, что возникновение ошибок будет происходить каскадно. Прежде чем переименовывать представление, следует хорошо продумать, к чему это может привести.*

➧**Пример 1. Создание представления Supplier.** Визуальными средствами создайте представление Supplier на основе таблиц Suppliers и Goods, определяемое следующей инструкцией:

```
CREATE VIEW Supplier AS
SELECT Title, Address, Telephone, Category, Mark, Price,
```

```
InWarehouse, Expected, MinimalStock, DeliveriesStopped
FROM Suppliers, Goods
WHERE Goods.CodeSuppliers=Suppliers.CodeSuppliers
```

Предложение WHERE говорит MS SQL Server как соединять строки таблиц между собой: столбец CodeSuppliers таблицы Goods является внешним ключом, который ссылается на столбец CodeSuppliers таблицы Suppliers. Оба столбца являются уникальными и не допускают неопределённых значений (NOT NULL), таким образом, значение столбца CodeSuppliers однозначно идентифицирует строку таблицы Suppliers.

Просмотрите результат выполнения одним из вышеописанных способов.

➡**Пример 2**. Визуальными средствами создайте представление MarkFromGoods на основе следующей инструкции:

```
SELECT Mark
FROM Goods
WHERE Supplier = (SELECT Supplier FROM Goods
WHERE Mark='Lexmark Z35')
```

➡**Пример 3.** Визуальными средствами создайте, представление Clients_SV_Trading на основе нижеследующего запроса, который выбирает название товара и фирмы, заказавшей какой-либо товар фирмы «SV-Trading».

```
SELECT Client, Goods
FROM Orders, Ordered
WHERE (Goods IN (SELECT Mark FROM Goods
        WHERE Supplier='SV-Trading'))
        AND Orders.CodeOrder=Ordered.CodeOrder
```

➡**Пример 4.** Визуальными средствами создайте представление ALL_Goods с использованием квантора ALL на основе нижеследующей инструкции:

```
SELECT Supplier, Mark, Price
FROM Goods
WHERE Price > ALL(SELECT Price FROM Goods
```

WHERE Supplier='CD-Life')

4 Контрольные вопросы

1) Объясните, чем отличается представление от запроса.
2) Для каких целей применяется представление?
3) Укажите порядок создания представления визуальными средствами.
4) Можно ли обновлять данные посредством представлений?
5) Можно ли создать представление на основе нескольких таблиц?

ЛИТЕРАТУРА

1. Solid Quality Learning Microsoft SQL Server 2005. Реализация и обслуживание. Учебный курс Microsoft / пер. с англ. – М.: «Русская Редакция», СПб.: «Питер», 2007. – 768 с.: ил.
2. Электронная документация по SQL Server 2008 [Электронный ресурс] – Режим доступа: http://msdn.microsoft.com/ru-ru/library/ms365325.aspx, свободный. – Загл. с экрана.
3. Библиотека MSDN [Электронный ресурс] – Режим доступа: http://msdn.microsoft.com/ru-ru/library/default.aspx, свободный. – Загл. с экрана.

ПРИЛОЖЕНИЯ

Приложение 1

Сценарий заполнения данными базы данных BOREI

```
/*------ Заполнение таблицы "Типы" ------*/
SET IDENTITY_INSERT Types ON;

INSERT INTO Types (CodeType, Category) VALUES (2, 'Принтеры');
INSERT INTO Types (CodeType, Category) VALUES (3, 'Материнские платы');
INSERT INTO Types (CodeType, Category) VALUES (4, 'Процессоры');
INSERT INTO Types (CodeType, Category) VALUES (5, 'Модули памяти');
INSERT INTO Types (CodeType, Category) VALUES (6, 'Видеокарты');
INSERT INTO Types (CodeType, Category) VALUES (7, 'Звуковые карты');
INSERT INTO Types (CodeType, Category)
VALUES (8, 'Устройства ввода и указания');
INSERT INTO Types (CodeType, Category)
 VALUES (9, 'Колонки, наушники, микрофоны');
INSERT INTO Types (CodeType, Category)
VALUES (10, 'Корпуса и блоки питания');
INSERT INTO Types (CodeType, Category) VALUES (11, 'Жесткие диски');
INSERT INTO Types (CodeType, Category) VALUES (12, 'Флоппи-дисководы');
INSERT INTO Types (CodeType, Category) VALUES (13, 'CD-ROM');
INSERT INTO Types (CodeType, Category) VALUES (14, 'Носители');
INSERT INTO Types (CodeType, Category) VALUES (15, 'Модемы');
INSERT INTO Types (CodeType, Category) VALUES (16, 'Сетевое оборудование');

SET IDENTITY_INSERT Types OFF;

/*------ Заполнение таблицы "Поставщики" ------*/
SET IDENTITY_INSERT Suppliers ON;

INSERT INTO Suppliers (CodeSuppliers, Title, AddressTo, Post, Address, City, IIndex,
Country, Telephone, Fax)
VALUES (3, 'BelSoft', 'Андрей Герасимов', 'Представитель', '707 Оксфорд',
    'Анн-Арбор', '48104', 'США', '(313) 555-5753', '(313) 555-3349');
INSERT INTO Suppliers (CodeSuppliers, Title, AddressTo, Post, Address, City, IIndex,
Country, Telephone)
VALUES (4, 'Iven', 'Антон Сеткин', 'Главный менеджер', '9-8 Секимаи',
    'Токио', '100', 'Япония', '(03) 3555-5011');
INSERT INTO Suppliers (CodeSuppliers, Title, AddressTo, Post, Address, City, IIndex,
```

Country, Telephone)
VALUES (5, 'SV-Trading', 'Валерия Евенкова', 'Директор', '92 Сетсако',
'Осака', '545', 'Япония', '(06) 431-7877');
INSERT INTO Suppliers (CodeSuppliers, Title, AddressTo, Post, Address, City, Index,
Country, Telephone, Fax)
VALUES (6, 'CD-Life', 'Наталия Отока', 'Главный менеджер', '74 ул. Роз',
'Мельбурн', '3058', 'Австралия', '(03) 444-2343', '(03) 444-6588');
INSERT INTO Suppliers (CodeSuppliers, Title, AddressTo, Post, Address, City, Index,
Country, Telephone)
VALUES (7, 'Stepfor', 'Павел Фокин', 'Представитель', 'ул. Королевского пути, 29',
'Манчестер', 'M14 GSD', 'Великобритания', '(161) 555-4448');
INSERT INTO Suppliers (CodeSuppliers, Title, AddressTo, Post, Address, City, Index,
Country, Telephone, Fax)
VALUES (8, 'B.S.T.Group', 'Евгений Шаматранов', 'Продавец', 'Каладоган 13',
'Гетеборг', 'S-345 67', 'Швеция', '031-987 65 43', '031-987 65 91');
INSERT INTO Suppliers (CodeSuppliers, Title, AddressTo, Post, Address, City, Index,
Country, Telephone)
VALUES (9, 'Biocom', 'Вячеслав Путеев', 'Главный менеджер',
'ул. Американская 12.890', 'Сан-Паулу', '5442', 'Бразилия', '(11) 555-4640');
INSERT INTO Suppliers (CodeSuppliers, Title, AddressTo, Post, Address, City, Index,
Country, Telephone)
VALUES (10, 'DAAS', 'Петр Моргунов', 'Менеджер по продажам', 'Тверская 5',
'Москва', '101785', 'Россия', '(095) 998-4510');
INSERT INTO Suppliers (CodeSuppliers, Title, AddressTo, Post, Address, City, Index,
Country, Telephone, Fax)
VALUES (11, 'Конструктив', 'Федор Куполов', 'Представитель', 'ул. Данте 75',
'Равенна', '48100', 'Италия', '(0544) 60323', '(0544) 60603');
INSERT INTO Suppliers (CodeSuppliers, Title, AddressTo, Post, Address, City, Index,
Country, Telephone)
VALUES (12, 'Виола-Сервис', 'Виктор Кухарчук', 'Главный менеджер',
'ул. Хатлевеген, 5', 'Сандвикен', '1320', 'Норвегия', '(0)2-953010');
INSERT INTO Suppliers (CodeSuppliers, Title, AddressTo, Post, Address, City, Index,
Country, Telephone)
VALUES (13, 'Твин-Медиа', 'Артем Столяров', 'Местный представитель',
'3400-8 Авеню', 'Бенд', '97101', 'США', '(503) 555-9931');
INSERT INTO Suppliers (CodeSuppliers, Title, AddressTo, Post, Address, City, Index,
Country, Telephone)
VALUES (14, 'Q-Senter', 'Александр Осипенко', 'Представитель',
'ул. Бровайдер, 231', 'Стокгольм', 'S-123 45', 'Швеция', '08-123 45 67');
INSERT INTO Suppliers (CodeSuppliers, Title, AddressTo, Post, Address, City, Index,
Country, Telephone, Fax)
VALUES (15, 'Техинтерторг', 'Дарья Борщева', 'Агент по продажам',

```
 'Частный Департамент', 'Бостон', '02134', 'США', '(617) 555-3267', '(617) 555-3389');
INSERT INTO Suppliers (CodeSuppliers, Title, AddressTo, Post, Address, City, Index,
Country, Telephone)
VALUES (16, 'Ситипринт', 'Инна Ринкевич', 'Совладелец', 'ул. Серангун, 471',
    'Сингапур', '0512', 'Сингапур', '555-8787');
INSERT INTO Suppliers (CodeSuppliers, Title, AddressTo, Post, Address, City, Index,
Country, Telephone, Fax)
VALUES (17, 'Юнити Сервис', 'Алексей Жолнерович', 'Менеджер по продажам',
 'ул. Лингбисилд', 'Лингби', '2800', 'Дания', '43844108', '43844115');
INSERT INTO Suppliers (CodeSuppliers, Title, AddressTo, Post, Address, City, Index,
Country, Telephone)
VALUES (18, 'ТК Сервис', 'Юрий Бартошек', 'Менеджер по продажам',
 'ул. Войрон, 22', 'Монсо', '71300', 'Франция', '85-57-00-07');
INSERT INTO Suppliers (CodeSuppliers, Title, AddressTo, Post, Address, City, Index,
Country, Telephone, Fax)
VALUES (19, 'Астлайн', 'Руслан Сидюк', 'Бухгалтер', 'ул. Чессер, 148',
    'Сте-Хиацинте', 'J2S 7S8', 'Канада', '(514) 555-2955', '(514) 555-2921');

SET IDENTITY_INSERT Suppliers OFF;

/*----- Заполнение таблицы "Клиенты" ------*/
SET IDENTITY_INSERT Clients ON;

INSERT INTO Clients (CodeClient, Title, AddressTo, Post, Address, City, Index,
Country, Telephone, Fax)
VALUES ('1', 'Омикс', 'Андрей Ханавин', 'Координатор', 'ул. Бергуса, 8',
    'Лулео', 'S-958 22', 'Швеция', '0921-12 34 65', '0921-12 34 67');
INSERT INTO Clients (CodeClient, Title, AddressTo, Post, Address, City, Index,
Country, Telephone)
VALUES ('2', 'МАП Инфо', 'Виктория Асворд', 'Представитель', 'ул. Цикровая',
    'Лондон', 'EC2 5N4T', 'Великобритания', '(171) 555-1212');
INSERT INTO Clients (CodeClient, Title, AddressTo, Post, Address, City, Index,
Country, Telephone, Fax)
VALUES ('3', 'Белкантон', 'Андрей Карпухин', 'Главный менеджер',
'ул. Гарсиа, 9993', 'Мехико', '05022', 'Мексика', '(5) 555-3392', '(5) 555-7293');
INSERT INTO Clients (CodeClient, Title, AddressTo, Post, Address, City, Index,
Country, Telephone)
VALUES ('4', 'Дамодара-Сервис', 'Лидия Куласва', 'Совладелец', 'ул. Эдальго, 29',
    'Берн', '3012', 'Швейцария', '0452-076545');
INSERT INTO Clients (CodeClient, Title, AddressTo, Post, Address, City, Index,
Country, Telephone, Fax)
VALUES ('5', 'E-Life', 'Роланд Мендель', 'Менеджер по продажам',
```

'ул. Кировская, 6', 'Грасс', '8010', 'Австрия', '7675-3425', '7675-3426');
INSERT INTO Clients (CodeClient, Title, AddressTo, Post, Address, City, Index, Country, Telephone)
VALUES ('6', 'IMC Computers', 'Мария Ларсон', 'Совладелец', 'ул. Кергатан, 24',
'Брекке', 'S-844 67', 'Швеция', '0695-34 67 21');
INSERT INTO Clients (CodeClient, Title, AddressTo, Post, Address, City, Index, Country, Telephone, Fax)
VALUES ('7', 'Версa', 'Питер Франкен', 'Главный менеджер', 'Берлинская пл., 43',
'Мюнхен', '80805', 'Германия', '089-0877310', '089-0877451');
INSERT INTO Clients (CodeClient, Title, AddressTo, Post, Address, City, Index, Country, Telephone, Fax)
VALUES ('8', 'Comtris', 'Мария Хосе', 'Совладелец', 'ул. Палос, 5S',
'Каракас', '1081', 'Венесуэла', '(2) 283-2951', '(2) 283-3397');
INSERT INTO Clients (CodeClient, Title, AddressTo, Post, Address, City, Index, Country, Telephone, Fax)
VALUES ('9', 'Net Line', 'Марио Понтес', 'Бухгалтер', 'ул. Ракко, 67',
'Рио-де-Жанейро', '05454-876', 'Бразилия', '(21) 555-0091', '(21) 555-8765');
INSERT INTO Clients (CodeClient, Title, AddressTo, Post, Address, City, Index, Country, Telephone, Fax)
VALUES ('10', 'NTTs', 'Карлос Хемандос', 'Представитель', 'ул. Карлос, 22',
'Сан-Кристобаль', '5022', 'Венесуэла', '(5) 555-1340', '(5) 555-1948');
INSERT INTO Clients (CodeClient, Title, AddressTo, Post, Address, City, Country, Telephone, Fax)
VALUES ('11', 'ИнтеллектС', 'Патрисия Кемма', 'Ученик продавца',
'Джонстоун шоссе, 8', 'Корк', 'Ирландия', '2967 542', '2967 3333');
INSERT INTO Clients (CodeClient, Title, AddressTo, Post, Address, City, Index, Country, Telephone, Fax)
VALUES ('12', 'ZS', 'Максим Анищенко', 'Бухгалтер', 'Боливарская, 52', 'Баркисимето', '3508', 'Венесуэла', '(9) 331-6954', '(9) 331-7256');
INSERT INTO Clients (CodeClient, Title, AddressTo, Post, Address, City, Index, Country, Telephone, Fax)
VALUES ('13', 'Медиа-софт', 'Джовани Ровелли', 'Главный менеджер',
'ул. Людовика, 22', 'Бергамо', '24100', 'Италия', '035-640230', '035-640231');
INSERT INTO Clients (CodeClient, Title, AddressTo, Post, Address, City, Index, Country, Telephone, Fax)
VALUES ('14', 'Эликон-М', 'Александр Боровик', 'Представитель',
'ул. Беринговая, 2743', 'Анкоридж', '99508', 'США', '(907) 555-7584', '(907) 555-2880');
INSERT INTO Clients (CodeClient, Title, AddressTo, Post, Address, City, Index, Country, Telephone)
VALUES ('15', 'Stop', 'Игнат Довидовский', 'Бухгалтер', 'Гачеретрасс, 10',
'Кюневальд', '01307', 'Германия', '0372-035188');

```
INSERT INTO Clients (CodeClient, Title, AddressTo, Post, Address, City, Index,
Country, Telephone, Fax)
VALUES ('16', 'Соло', 'Андрей Савельев', 'Ученик продавца',
'ул. Провинциальная, 124', 'Реджио-Эмилио', '42100', 'Италия', '0522-556721', '0522-
556722');
INSERT INTO Clients (CodeClient, Title, AddressTo, Post, Address, City, Index,
Country, Telephone, Fax)
VALUES ('17', 'Olymp', 'Юрий Макаров', 'Бухгалтер', 'ул. Виа, 1', 'Мадрид', '28001',
'Испания', '(91) 745 6200', '(91) 745 6210');
INSERT INTO Clients (CodeClient, Title, AddressTo, Post, Address, City, Index,
Country, Telephone, Fax)
VALUES ('18', 'Oki', 'Александр Гронский', 'Представитель', 'ул. Кастро, 414',
'Сан-Пауло', '05634-030', 'Бразилия', '(11) 555-2167', '(11) 555-2168');

SET IDENTITY_INSERT Clients OFF;
/*------ Заполнение таблицы "Доставка" ------*/
INSERT INTO Deliveries (CodeDelivery, Title, Telephone)
VALUES (1, 'Ространс', '(017) 972-9831');
INSERT INTO Deliveries (CodeDelivery, Title, Telephone)
VALUES (2, 'Почта', '(017) 124-3199');
INSERT INTO Deliveries (CodeDelivery, Title, Telephone)
VALUES (3, 'Иное', '(017) 211-9931');


/*------ Заполнение таблицы "Сотрудники" ------*/
SET IDENTITY_INSERT Employees ON;

INSERT INTO Employees (CodeEmployee, SurName, Name, Post, Reference,
DateBirth, DateHiring,
        Address, City, Index, Country, HomeTelephone, Additional, Submits)
VALUES (2, 'Белова', 'Мария', 'Представитель', 'г-жа.', '08.12.1968', '01.05.1992',
'ул. Нефтяников, 14-4', 'Минск', '122981', 'Беларусь', '(017) 555-9857', '124-5467',
'Новиков, Павел');
INSERT INTO Employees (CodeEmployee, SurName, Name, Post, Reference,
DateBirth, DateHiring,
        Address, City, Index, Country, HomeTelephone, Additional, Submits)
VALUES (8, 'Крылова', 'Анна', 'Внутренний координатор', 'г-жа.', '09.01.1958',
'05.03.1994', 'ул. Лесная, 12-456',
'Минск', '105001', 'Беларусь', '(017) 555-1189', '124-2344', 'Кротов, Андрей');

SET IDENTITY_INSERT Employees OFF;
```

```
/*------ Заполнение таблицы "Товары" ------*/
SET IDENTITY_INSERT Goods ON;

INSERT INTO Goods (CodeGoods, Mark, CodeSuppliers, CodeType, Price,
InWarehouse, Expected, MinimalStock)
VALUES (1, 'Genius SP-G06', 1, 9, 9.00, 20, 10, 10);
INSERT INTO Goods (CodeGoods, Mark, CodeSuppliers, CodeType, Price,
InWarehouse, Expected)
VALUES (2, 'Наушники Dialog M-750HV+микрофон', 8, 9, 9.00, 15, 10);
INSERT INTO Goods (CodeGoods, Mark, CodeSuppliers, CodeType, Price,
InWarehouse, Expected, MinimalStock)
VALUES (3, 'ATX 2.03 300W', 19, 10, 35.00, 15, 5,10);
INSERT INTO Goods (CodeGoods, Mark, CodeSuppliers, CodeType, Price,
InWarehouse)
VALUES (4, 'ATX Midi Tower 350W', 7, 10, 55.00, 5);
INSERT INTO Goods (CodeGoods, Mark, CodeSuppliers, CodeType, Price,
InWarehouse, Expected, MinimalStock)
VALUES (5, 'Блок питания ATX 235/250/300W', 5, 10, 14.00, 20, 10, 5);
INSERT INTO Goods (CodeGoods, Mark, CodeSuppliers, CodeType, Price,
InWarehouse, Expected, MinimalStock)
VALUES (6, 'HDD Maxtor 30Gb', 16, 11, 69.00, 10, 2, 8);
INSERT INTO Goods (CodeGoods, Mark, CodeSuppliers, CodeType, Price)
VALUES (7, 'HDD Seagate Baracuda 4 40Gb', 1, 11, 87.00);
INSERT INTO Goods (CodeGoods, Mark, CodeSuppliers, CodeType, Price,
InWarehouse)
VALUES (8, 'HDD IBM 60Gb', 13, 11, 79.00, 5);
INSERT INTO Goods (CodeGoods, Mark, CodeSuppliers, CodeType, Price,
InWarehouse, Expected, MinimalStock)
VALUES (9, 'TEAC', 4, 12, 9.00, 15, 10, 10);
INSERT INTO Goods (CodeGoods, Mark, CodeSuppliers, CodeType, Price)
VALUES (10, 'Samsung', 14, 12, 13.00);
INSERT INTO Goods (CodeGoods, Mark, CodeSuppliers, CodeType, Price,
InWarehouse, Expected)
VALUES (11, 'Samsung 52x', 8, 13, 23.00, 5, 5);
INSERT INTO Goods (CodeGoods, Mark, CodeSuppliers, CodeType, Price,
InWarehouse)
VALUES (12, 'TEAC 52x CD-522E', 3, 13, 42.00, 7);
INSERT INTO Goods (CodeGoods, Mark, CodeSuppliers, CodeType, Price,
InWarehouse, Expected, MinimalStock)
VALUES (13, 'CD-RW/ROM TEAC', 11, 13, 25.00, 20, 10, 10);
INSERT INTO Goods (CodeGoods, Mark, CodeSuppliers, CodeType, Price,
InWarehouse, Expected, MinimalStock)
```

```
VALUES (14, 'Verbatim', 3, 14, 0.28, 300, 100, 50);
INSERT INTO Goods (CodeGoods, Mark, CodeSuppliers, CodeType, Price,
InWarehouse, Expected, MinimalStock)
VALUES (15, 'CD-R TDK 12x', 19, 14, 0.62, 200, 100, 50);
INSERT INTO Goods (CodeGoods, Mark, CodeSuppliers, CodeType, Price,
InWarehouse, Expected)
VALUES (16, 'CD-R/RW Verbatim 650/700 Mb', 9, 14, 0.98, 200, 100);
INSERT INTO Goods (CodeGoods, Mark, CodeSuppliers, CodeType, Price,
InWarehouse, Expected)
VALUES (17, '3COM USR Zyxel', 4, 15, 18.00, 10, 5);
INSERT INTO Goods (CodeGoods, Mark, CodeSuppliers, CodeType, Price,
InWarehouse)
VALUES (18, 'ACORP 56EMS, USB', 5, 15, 40.00, 5);
INSERT INTO Goods (CodeGoods, Mark, CodeSuppliers, CodeType, Price,
InWarehouse, Expected)
VALUES (19, '3COM SOHO 100TX', 10, 16, 22.00, 15, 5);
INSERT INTO Goods (CodeGoods, Mark, CodeSuppliers, CodeType, Price,
InWarehouse, Expected)
VALUES (20, 'HUB 10/100Mbit', 3, 16, 32.00, 10, 5);
INSERT INTO Goods (CodeGoods, Mark, CodeSuppliers, CodeType, Price,
InWarehouse, Expected, MinimalStock)
VALUES (21, 'Switch 10/100Mbit', 13, 16, 32.00, 10, 10, 5);
INSERT INTO Goods (CodeGoods, Mark, CodeSuppliers, CodeType, Price,
InWarehouse, Expected, MinimalStock)
VALUES (22, 'Кабель "витая пара" UTP cat.5', 4, 16, 0.18, 100, 100, 20);
INSERT INTO Goods (CodeGoods, Mark, CodeSuppliers, CodeType, Price,
InWarehouse, MinimalStock, DeliveriesStopped)
VALUES (23, 'Canon LBP810', 3, 2, 185.00, 39, 5, 'YES');
INSERT INTO Goods (CodeGoods, Mark, CodeSuppliers, CodeType, Price,
InWarehouse, MinimalStock)
VALUES (24, 'Accorp VIA266i815D', 7, 3, 61.00, 29, 10);
INSERT INTO Goods (CodeGoods, Mark, CodeSuppliers, CodeType, Price,
DeliveriesStopped)
VALUES (25, 'TNT2 M64 32Mb', 7, 6, 28.00, 'YES');
INSERT INTO Goods (CodeGoods, Mark, CodeSuppliers, CodeType, Price,
InWarehouse, MinimalStock)
VALUES (26, 'DDR 128Mb', 9, 5, 38.00, 34, 25);
INSERT INTO Goods (CodeGoods, Mark, CodeSuppliers, CodeType, Price,
InWarehouse, DeliveriesStopped)
VALUES (27, 'Samsung 550B', 10, 1, 150.00, 20, 'YES');
INSERT INTO Goods (CodeGoods, Mark, CodeSuppliers, CodeType, Price,
InWarehouse, MinimalStock)
```

```
VALUES (28, 'Gigabyte GA-60XTA', 11, 3, 64.00, 76, 30);
INSERT INTO Goods (CodeGoods, Mark, CodeSuppliers, CodeType, Price,
InWarehouse, DeliveriesStopped)
VALUES (29, 'Creative Labs SB Life', 12, 7, 9.00, 26, 'YES');
INSERT INTO Goods (CodeGoods, Mark, CodeSuppliers, CodeType, Price,
InWarehouse, MinimalStock)
VALUES (30, 'Mouse Logitech B69', 13, 8, 15.00, 10, 15);
INSERT INTO Goods (CodeGoods, Mark, CodeSuppliers, CodeType, Price,
InWarehouse)
VALUES (31, 'Epson EPL520', 6, 2, 90.00, 76);
INSERT INTO Goods (CodeGoods, Mark, CodeSuppliers, CodeType, Price,
InWarehouse, MinimalStock)
VALUES (32, 'Celeron 1000MHz', 15, 4, 51.00, 26, 15);
INSERT INTO Goods (CodeGoods, Mark, CodeSuppliers, CodeType, Price, Expected,
MinimalStock)
VALUES (33, 'Intel P4 1700MHz Box', 14, 4, 137.00, 70, 20);
INSERT INTO Goods (CodeGoods, Mark, CodeSuppliers, CodeType, Price,
InWarehouse, Expected, MinimalStock)
VALUES (34, 'AMD Athlon 1333MHz', 6, 4, 53.00, 9, 40, 25);
INSERT INTO Goods (CodeGoods, Mark, CodeSuppliers, CodeType, Price,
InWarehouse, MinimalStock)
VALUES (35, 'Samtron 76E', 16, 1, 180.00, 11, 15);
INSERT INTO Goods (CodeGoods, Mark, CodeSuppliers, CodeType, Price,
InWarehouse, MinimalStock)
VALUES (36, 'Коврики пластиковые', 17, 8, 0.50, 112, 20);
INSERT INTO Goods (CodeGoods, Mark, CodeSuppliers, CodeType, Price,
InWarehouse, MinimalStock)
VALUES (37, 'Chicony PS/2 820 2981', 19, 8, 6.00, 23, 30);
INSERT INTO Goods (CodeGoods, Mark, CodeSuppliers, CodeType, Price,
InWarehouse, DeliveriesStopped)
VALUES (38, 'SDRAM 128Mb', 18, 5, 19.00, 26, 'YES');
INSERT INTO Goods (CodeGoods, Mark, CodeSuppliers, CodeType, Price,
InWarehouse, MinimalStock)
VALUES (39, 'Asus A7S333', 6, 3, 86.00, 35, 30);
INSERT INTO Goods (CodeGoods, Mark, CodeSuppliers, CodeType, Price,
InWarehouse, MinimalStock)
VALUES (40, 'Creative Labs SB Life 5.1+FM radio', 2, 7, 39.00, 10, 10);
INSERT INTO Goods (CodeGoods, Mark, CodeSuppliers, CodeType, Price,
InWarehouse, Expected, MinimalStock)
VALUES (41, 'GeForce 2 MX 400 64Mb', 2, 6, 36.00, 10, 15, 24);
INSERT INTO Goods (CodeGoods, Mark, CodeSuppliers, CodeType, Price,
InWarehouse, Expected, MinimalStock)
```

```
VALUES (42, 'DIMM PC133 128Mb Micron', 2, 5, 15.00, 21, 10, 30);
INSERT INTO Goods (CodeGoods, Mark, CodeSuppliers, CodeType, Price,
InWarehouse, Expected, MinimalStock)
VALUES (43, 'AND Duron 1000MHz', 9, 4, 36.00, 22, 10, 10);
INSERT INTO Goods (CodeGoods, Mark, CodeSuppliers, CodeType, Price,
InWarehouse, MinimalStock)
VALUES (44, 'Lexmark Z35', 5, 2, 62.00, 12, 25);
INSERT INTO Goods (CodeGoods, Mark, CodeSuppliers, CodeType, Price,
InWarehouse, DeliveriesStopped)
VALUES (45, 'TV-tuner Aver MediaTV Studio', 4, 6, 75.00, 29, 'YES');

SET IDENTITY_INSERT Goods OFF;

/*----- Заполнение таблицы "Заказы" ------*/
SET IDENTITY_INSERT Orders ON;

INSERT INTO Orders (CodeOrder, CodeClient, CodeEmployee, DateAccomodation,
DatePurpose, DateExecution, CodeDelivery, CostDelivery, TitleAddressee,
AddressAddressee, CityAddressee, IndexAddressee, CountryAddressee)
VALUES(11010, '16', 2, '01.05.1992', '05.03.1994', '01.05.1992', 2, 28.70, 'Соло',
    'Тачерстрасс, 10', 'Лондон', 'WX3 6FW', 'Великобритания');
INSERT INTO Orders (CodeOrder, CodeClient, CodeEmployee, DateAccomodation,
DatePurpose, DateExecution, CodeDelivery, CostDelivery, TitleAddressee,
        AddressAddressee, CityAddressee, IndexAddressee, CountryAddressee)
VALUES(11011, '6', 8, '01.05.1992', '05.03.1994', '01.05.1992', 1, 1.21,
'IMC Computers', 'ул. Монте, 34', 'Берлин', '12209', 'Германия');
INSERT INTO Orders (CodeOrder, CodeClient, CodeEmployee, DateAccomodation,
DatePurpose, DateExecution, CodeDelivery, CostDelivery, TitleAddressee,
        AddressAddressee, CityAddressee, IndexAddressee, CountryAddressee)
VALUES(11012, '7', 2, '01.05.1992', '05.03.1994', '01.05.1992', 3, 243.00, 'Верса',
    'ул. Джардем, 32', 'Мюнхен', '80805', 'Германия');
INSERT INTO Orders (CodeOrder, CodeClient, CodeEmployee, DateAccomodation,
DatePurpose, DateExecution, CodeDelivery, CostDelivery, TitleAddressee,
        AddressAddressee, CityAddressee, IndexAddressee, CountryAddressee)
VALUES(11013, '17', 8, '01.05.1992', '05.03.1994', '01.05.1992', 1, 33.00, 'Olymp',
    'ул.Каталана, 23', 'Мадрид', '28001', 'Испания');
INSERT INTO Orders (CodeOrder, CodeClient, CodeEmployee, DateAccomodation,
DatePurpose,DateExecution, CodeDelivery, CostDelivery, TitleAddressee,
        AddressAddressee, CityAddressee, IndexAddressee, CountryAddressee)
VALUES(11014, '14', 2, '01.05.1992', '05.03.1994', '01.05.1992', 3, 23.60, 'Эликон-М',
    'ул. Веговая, 7', 'Бранденбург', '14776', 'Германия');
INSERT INTO Orders (CodeOrder, CodeClient, CodeEmployee, DateAccomodation,
```

```
DatePurpose, DateExecution, CodeDelivery, CostDelivery, TitleAddressee,
        AddressAddressee, CityAddressee, IndexAddressee, CountryAddressee)
VALUES(11015, '11', 8, '01.05.1992', '05.03.1994', '01.05.1992', 2, 4.62, 'ИнтеллектС',
    'Бразильская пл., 442', 'Ставерен', '4110', 'Норвегия');
INSERT INTO Orders (CodeOrder, CodeClient, CodeEmployee, DateAccomodation,
DatePurpose, DateExecution, CodeDelivery, CostDelivery, TitleAddressee,
        AddressAddressee, CityAddressee, IndexAddressee, CountryAddressee)
VALUES(11016, '7', 2, '01.05.1992', '05.03.1994', '01.05.1992', 2, 33.80, 'Версa',
    'ул. Джардем, 32', 'Мюнхен', '80805', 'Германия');
INSERT INTO Orders (CodeOrder, CodeClient, CodeEmployee, DateAccomodation,
DatePurpose, DateExecution, CodeDelivery, CostDelivery, TitleAddressee,
        AddressAddressee, CityAddressee, IndexAddressee, CountryAddressee)
VALUES(11017, '5', 8, '01.05.1992', '05.03.1994', '01.05.1992', 2, 754.00, 'E-Life',
    'ул. Мерхеместа, 369', 'Юджин', '97403', 'США');
INSERT INTO Orders (CodeOrder, CodeClient, CodeEmployee, DateAccomodation,
DatePurpose, DateExecution, CodeDelivery, CostDelivery, TitleAddressee,
        AddressAddressee, CityAddressee, IndexAddressee, CountryAddressee)
VALUES(11018, '16', 2, '01.05.1992', '05.03.1994', '01.05.1992', 2, 11.70, 'Соло',
    'Тачерстрасс, 10', 'Лондон', 'WX3 6FW', 'Великобритания');
INSERT INTO Orders (CodeOrder, CodeClient, CodeEmployee, DateAccomodation,
DatePurpose, DateExecution, CodeDelivery, CostDelivery, TitleAddressee,
        AddressAddressee, CityAddressee, IndexAddressee, CountryAddressee)
VALUES(11019, '3', 8, '01.05.1992', '05.03.1994', '01.05.1992', 3, 3.17, 'Белкантон',
    'ул. Карлос, 22', 'Буэнос-Айрес', '1010', 'Аргентина');
INSERT INTO Orders (CodeOrder, CodeClient, CodeEmployee, DateAccomodation,
DatePurpose, DateExecution, CodeDelivery, CostDelivery, TitleAddressee,
        AddressAddressee, CityAddressee, IndexAddressee, CountryAddressee)
VALUES(11020, '9', 8, '01.05.1992', '05.03.1994', '01.05.1992', 2, 43.30, 'Net Line',
    'ул. Кэш, 321', 'Сан-Пауло', '05432-043', 'Бразилия');

SET IDENTITY_INSERT Orders OFF;

/*----- Заполнение таблицы "Заказано" ------*/
INSERT INTO Ordered (CodeOrder, CodeGoods, Amount, Discount)
VALUES (11010, 32, 5, 0.15);
INSERT INTO Ordered (CodeOrder, CodeGoods, Amount)
VALUES (11010, 44, 4);
INSERT INTO Ordered (CodeOrder, CodeGoods, Amount)
VALUES (11011, 16, 30);
INSERT INTO Ordered (CodeOrder, CodeGoods, Amount)
VALUES (11011, 42, 8);
INSERT INTO Ordered (CodeOrder, CodeGoods, Amount)
```

```
VALUES (11011, 43, 2);
INSERT INTO Ordered (CodeOrder, CodeGoods, Amount)
VALUES (11012, 9, 14);
INSERT INTO Ordered (CodeOrder, CodeGoods, Amount, Discount)
VALUES (11012, 26, 9, 0.05);
INSERT INTO Ordered (CodeOrder, CodeGoods, Amount)
VALUES (11013, 7, 9);
INSERT INTO Ordered (CodeOrder, CodeGoods, Amount)
VALUES (11013, 21, 10);
INSERT INTO Ordered (CodeOrder, CodeGoods, Amount, Discount)
VALUES (11014, 2, 16, 0.1);
INSERT INTO Ordered (CodeOrder, CodeGoods, Amount, Discount)
VALUES (11015, 6, 2, 0.15);
INSERT INTO Ordered (CodeOrder, CodeGoods, Amount)
VALUES (11016, 16, 40);
INSERT INTO Ordered (CodeOrder, CodeGoods, Amount)
VALUES (11017, 12, 12);
INSERT INTO Ordered (CodeOrder, CodeGoods, Amount)
VALUES (11018, 14, 80);
INSERT INTO Ordered (CodeOrder, CodeGoods, Amount, Discount)
VALUES (11019, 25, 7, 0.25);
INSERT INTO Ordered (CodeOrder, CodeGoods, Amount, Discount)
VALUES (11020, 30, 6, 0.15);
```

Сценарий обновления данных в базе данных BOREI

```
UPDATE Goods SET Category=(SELECT Category FROM Types WHERE
Goods.CodeType=Types.CodeType);

UPDATE Orders SET Client=(SELECT Title FROM Clients WHERE
Orders.CodeClient=Clients.CodeClient);

UPDATE Orders SET Employee=(SELECT Surname + ', ' + Name FROM Employees
                WHERE Orders.CodeEmployee=Employees.CodeEmployee);

UPDATE Orders SET Orders.Delivery=(SELECT Title FROM Deliveries
                    WHERE Orders.CodeDelivery=Deliveries.CodeDelivery);

UPDATE Ordered SET Ordered.Goods=(SELECT Mark FROM Goods WHERE
Ordered.CodeGoods=Goods.CodeGoods);

UPDATE Ordered SET Ordered.Price=(SELECT Price FROM Goods WHERE
Ordered.CodeGoods=Goods.CodeGoods);
```

Учебное издание

БАЗЫ ДАННЫХ

Лабораторный практикум
для студентов специализации 1-40 01 02-01
«Информационные технологии в производстве и управлении»

Составители:
КОЧУРОВ Вадим Александрович
ГЕРМАН Юлия Олеговна

Подписано в печать 26.11.2010.
Формат 60×84 $^{1}/_{16}$. Бумага офсетная.
Отпечатано на ризографе. Гарнитура Таймс.
Усл. печ. л. 5,23. Уч.-изд. л. 4,09. Тираж 100. Заказ 600.

Издатель и полиграфическое исполнение:
Белорусский национальный технический университет.
ЛИ № 02330/0494349 от 16.03.2009.
Проспект Независимости, 65. 220013, Минск.