

Лабораторная работа 1

Ознакомление с основными особенностями СУБД Microsoft SQL Server 2005 и возможностями интегрированной среды Microsoft SQL Server Management Studio. Создание базы данных и объектов базы данных

Описание задачи, рассматриваемой в лабораторном практикуме

Некоторая фирма приобретает товары у различных поставщиков (как юридических, так и физических лиц). Приобретение товаров осуществляется партиями и оформляется в виде договоров на поставку. Каждый договор на поставку товара имеет уникальный номер и может быть заключен только с одним поставщиком. В документах по каждому договору для каждого товара указываются: наименование, размер поставленной партии и цена (в грн.).

Для хранения и обработки подобной информации средствами СУБД Microsoft SQL Server 2005 необходимо создать базу данных.

ВЫПОЛНЕНИЕ РАБОТЫ

I. Создание базы данных

1. Создать на диске компьютера (D:, E: и т.п.) каталог с произвольным именем (например, E:\LABMSSQL)
2. Запустить Microsoft SQL Server Management Studio, для чего:
 - ♦ в панели задач выбрать пункт Microsoft SQL Server 2005;
 - ♦ выбрать подпункт SQL Server Management Studio;
 - ♦ в окне подключения (рисунок 1.1) нажать кнопку Connect;



Рисунок 1.1

3. После появления на экране среды Microsoft SQL Server Management Studio в окне Object Explorer выбрать пункт Databases, нажать правую кнопку мыши и в появившемся меню выбрать пункт New Database.... В результате на экране появится окно, позволяющее ввести основные параметры новой базы данных. Необходимо ввести имя новой базы данных – delivery и определить место размещения файлов - E:\LABMSSQL (рисунок 1.2). После ввода данных нажать кнопку ОК. Новая база данных появится в списке баз данных в окне Object Explorer (рисунок 1.3).

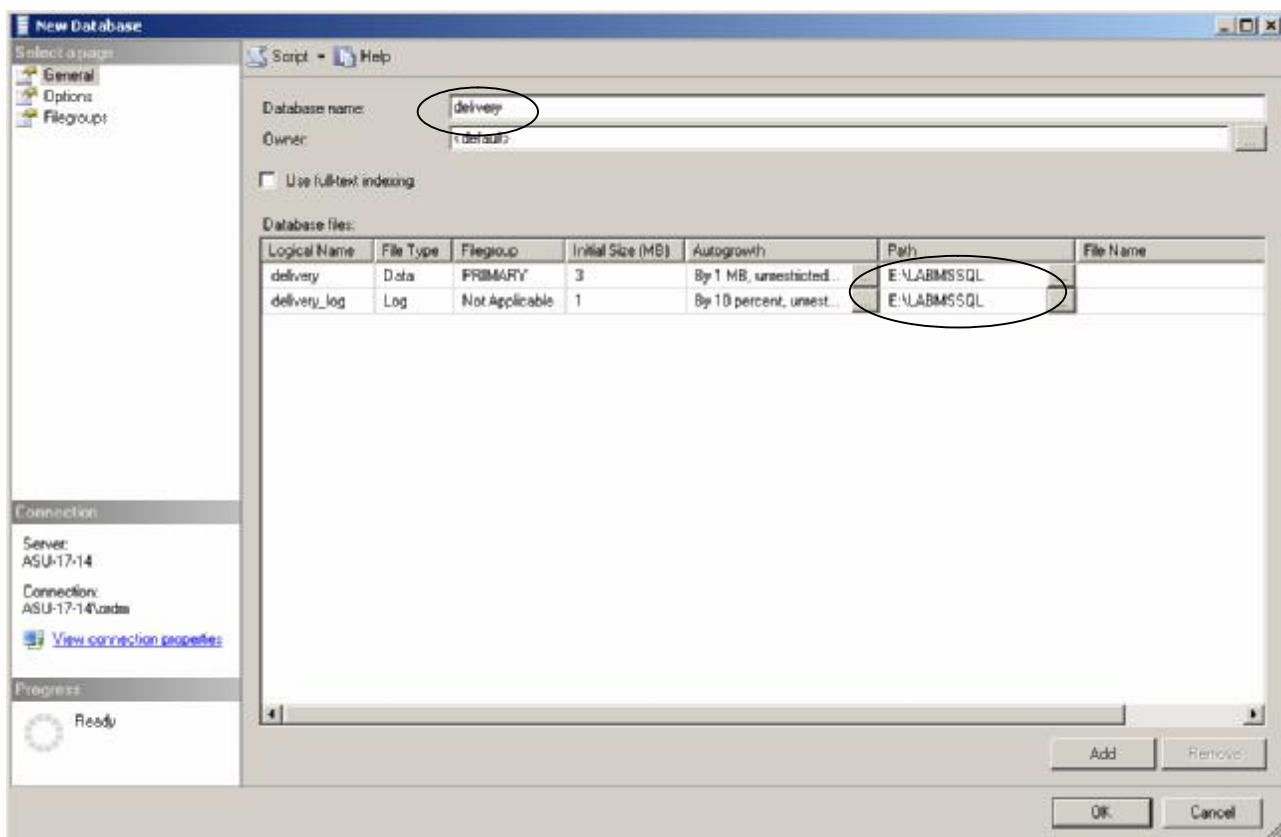


Рисунок 1.2

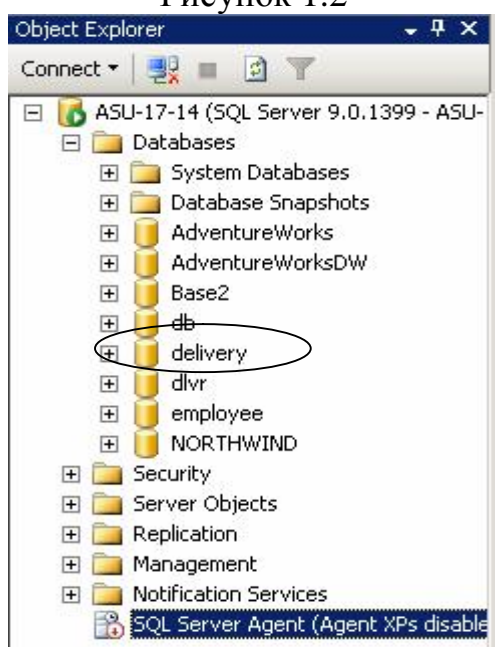


Рисунок 1.3

4. Выбрать созданную базу данных и раскрыть список ее объектов (рисунок 1.4).

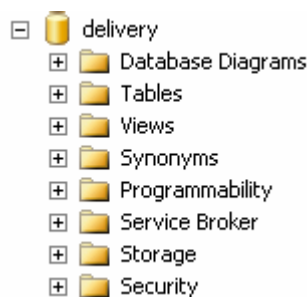


Рисунок 1.4

5. В списке объектов базы данных щелкнуть правой кнопкой мыши по пункту Tables и в появившемся меню выбрать пункт New Table.... Ввести поля новой таблицы (рисунок 1.5), определив при этом типы данных и ключевое поле (для этого нужно щелкнуть по полю правой кнопкой мыши и выбрать в меню соответствующий пункт (рисунок 1.6)).



	Column Name	Data Type	Allow Nulls
	КодПоставщика	int	<input type="checkbox"/>
	Адрес	text	<input type="checkbox"/>
	Примечание	text	<input checked="" type="checkbox"/>
			<input type="checkbox"/>

Рисунок 1.5

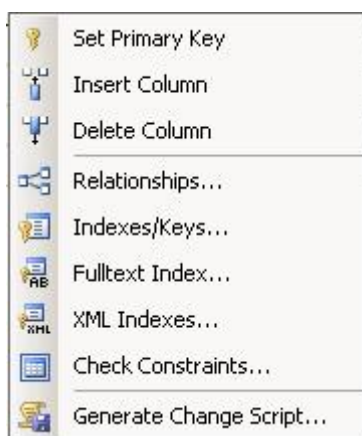


Рисунок 1.6

6. Закрыть вкладку со структурой новой таблицы. Сохранить новую таблицу с именем «Поставщики» (без кавычек).
7. Аналогично создать таблицы «ФизическиеЛица» и «ЮридическиеЛица». Их структуры приведены на рисунках 1.7 и 1.8 соответственно.

	Column Name	Data Type	Allow Nulls
►	КодПоставщика	int	<input type="checkbox"/>
	Фамилия	char(20)	<input type="checkbox"/>
	Имя	char(20)	<input type="checkbox"/>
	Отчество	char(20)	<input type="checkbox"/>
	НомерСвидетельства	char(10)	<input checked="" type="checkbox"/>
			<input type="checkbox"/>

Рисунок 1.7

	Column Name	Data Type	Allow Nulls
►	КодПоставщика	int	<input type="checkbox"/>
	Название	char(20)	<input type="checkbox"/>
	НалоговыйНомер	char(20)	<input checked="" type="checkbox"/>
	НомерСвидетельстваНДС	char(10)	<input checked="" type="checkbox"/>
			<input type="checkbox"/>

Рисунок 1.8

8. Создать таблицу «Договоры». Структура таблицы приведена на рисунке 1.9. Особенностью этой таблицы является то, что для поля «НомерДоговора» должно быть установлено свойство автоприращения (autoincrement) с начальным значением 1 и шагом изменения 1. Для этого в Microsoft SQL Server используется свойство Identity. Необходимо изменить значения свойства так, как показано на рисунке 1.10.

	Column Name	Data Type	Allow Nulls
►	НомерДоговора	int	<input type="checkbox"/>
	ДатаДоговора	datetime	<input checked="" type="checkbox"/>
	КодПоставщика	int	<input type="checkbox"/>
	Комментарий	text	<input checked="" type="checkbox"/>
			<input type="checkbox"/>

Рисунок 1.9

Column Properties	
<div> <div> <div></div> <div>A</div> <div>Z</div> <div>↓</div> </div> <div></div> </div>	
Has Non-SQL Server Subscriber	No
<input checked="" type="checkbox"/> Identity Specification	Yes
(Is Identity)	Yes
Identity Increment	1
Identity Seed	1
Indexable	Yes
Merge-published	No

Рисунок 1.10

9. Создать таблицу «Договоры». Структура таблицы приведена на рисунке 1.11. Особенностью таблицы является составной первичный ключ. Для его

создания нужно выделить ключевые поля (мышью, при прижатой клавише Shift) и затем определить их как ключевые

	Column Name	Data Type	Allow Nulls
▶	НомерДоговора	int	<input type="checkbox"/>
▶	Товар	char(20)	<input type="checkbox"/>
	Количество	decimal(5, 0)	<input type="checkbox"/>
	Цена	decimal(8, 2)	<input type="checkbox"/>
			<input type="checkbox"/>

Рисунок 1.11

10. В результате создания таблиц структура созданной базы данных будет иметь вид (рисунок 1.12). В том случае, если список таблиц не отображается, можно щелкнуть правой кнопкой мыши по имени базы данных и в появившемся меню выбрать пункт Refresh.

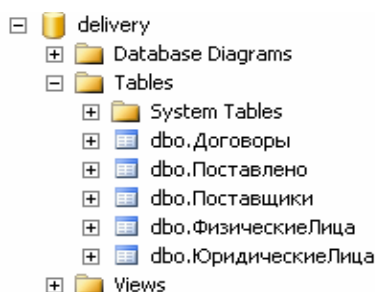


Рисунок 1.12

11. Теперь между созданными таблицами нужно установить связи. Это, в частности, можно сделать, используя визуальные средства. Для этого необходимо создать диаграмму базы данных. Для создания диаграммы нужно щелкнуть правой кнопкой мыши по пункту Database Diagrams (рисунок 1.12) и в появившемся меню выбрать пункт New Database Diagram. Затем нужно последовательно добавлять в состав диаграммы таблицы, выбирая их из списка и нажимая кнопку Add (рисунок 1.13)

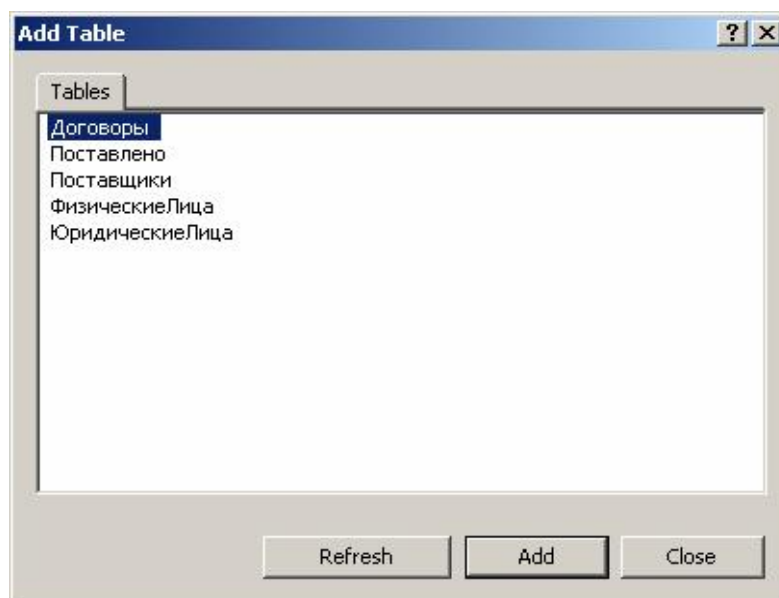


Рисунок 1.13

12. После включения таблиц в состав диаграммы нужно связать их ключевые поля. Для этого нужно выбрать с помощью мыши ключевое поле в родительской таблице и, не отпуская кнопку мыши, тянуть указатель мыши к дочерней таблице. В результате установления связи на экран будет выведено окно, отображающее имя связи и связываемые поля (рисунок 1.14). Этот пример отображает установление связи между таблицами «Поставщики» и «ЮридическиеЛица». Подтвердив параметры связи, пользователь затем может подтвердить или изменить параметры внешнего ключа и тип отношений ссылочной целостности (рисунок 1.15).

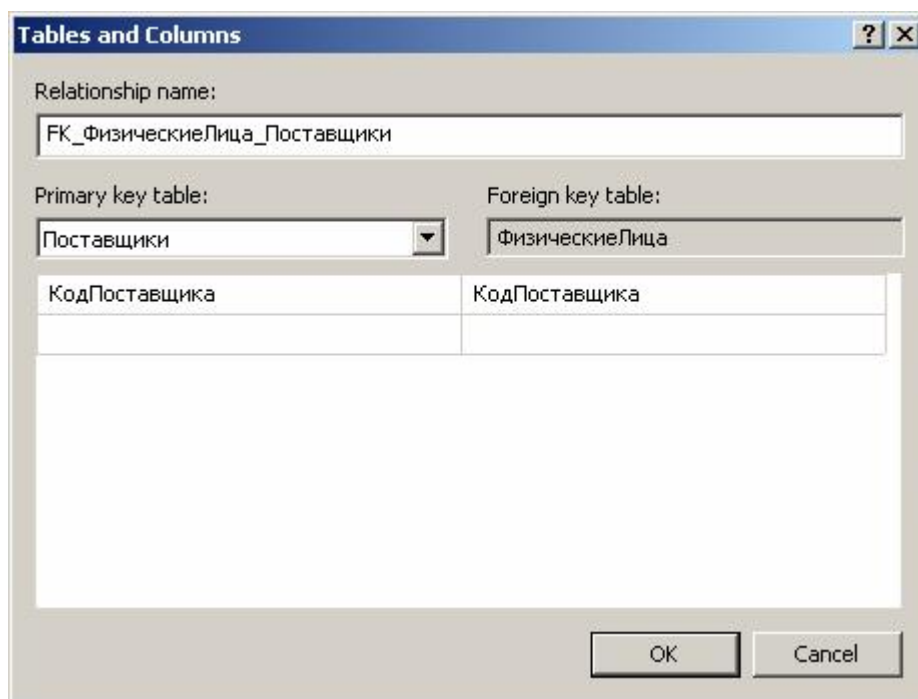


Рисунок 1.14

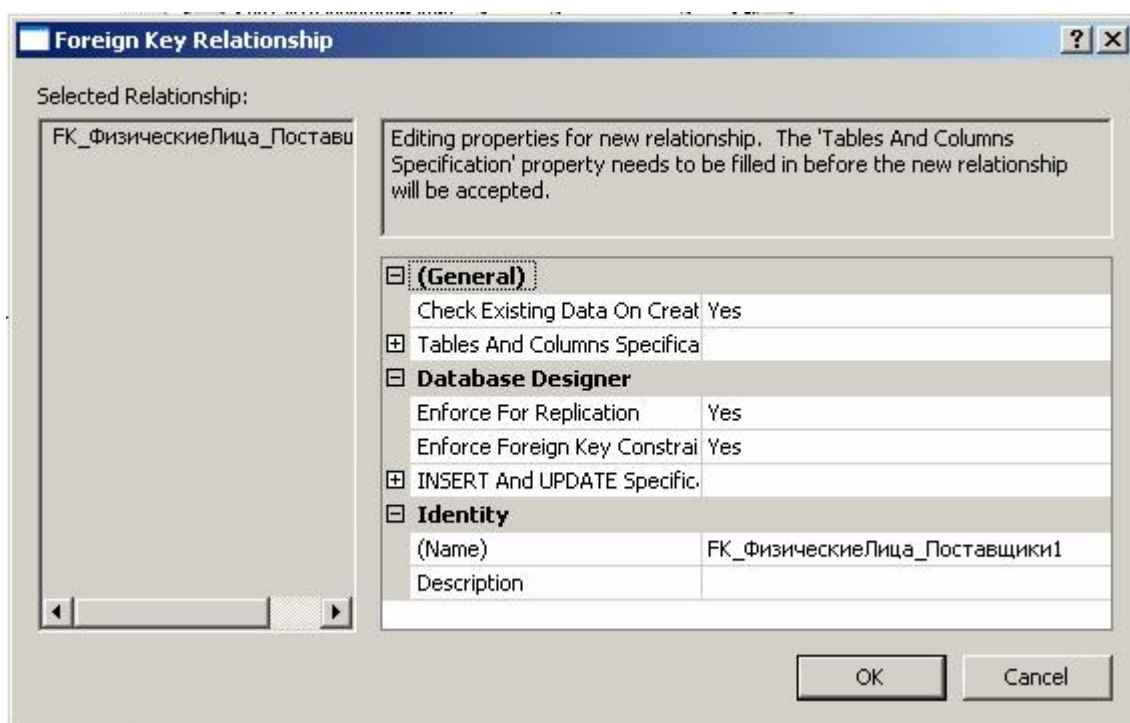


Рисунок 1.15

13. В результате установления связей между таблицами диаграмма может иметь вид (рисунок 1.15). Сформированную диаграмму можно закрыть и сохранить при этом с произвольным именем, например Diagram_0. Эта диаграмма появится в общем списке диаграмм базы данных.

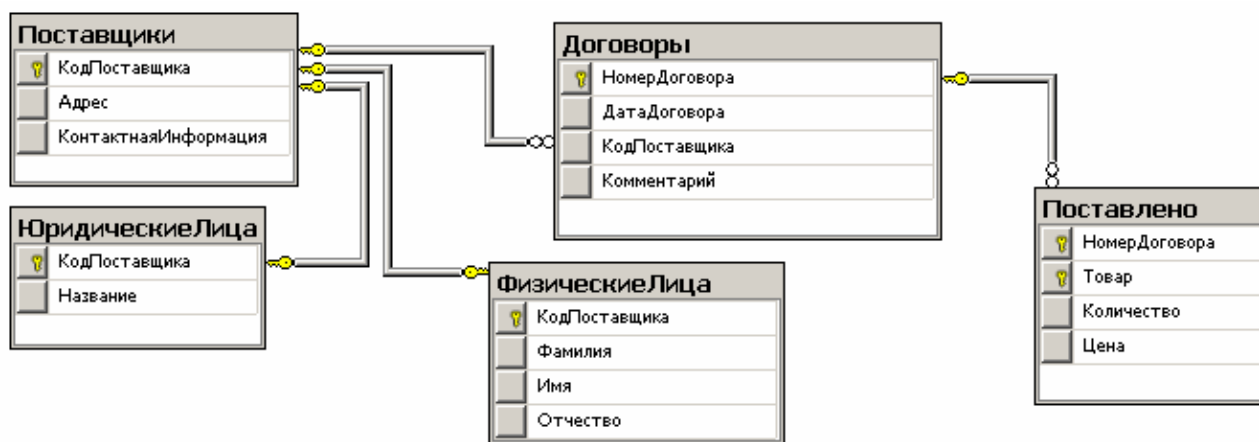


Рисунок 1.16

14. С помощью диаграммы базы данных можно изменять структуру таблиц, устанавливать связи, дополнительные свойства полей и т.д. Предположим, что для поле «Количество» и «Цена» таблицы «Поставлено» необходимо реализовать требования, состоящие в том, что данные, хранящиеся в этих полях, должны быть положительными. Для этого вновь откроем диаграмму, щелкнем правой кнопкой мыши по таблице «Поставлено» и в появившемся меню выберем пункт Check Constraints.... В появившемся окне нужно нажать кнопку Add и ввести выражение для контроля и название (рисунок 1.17).

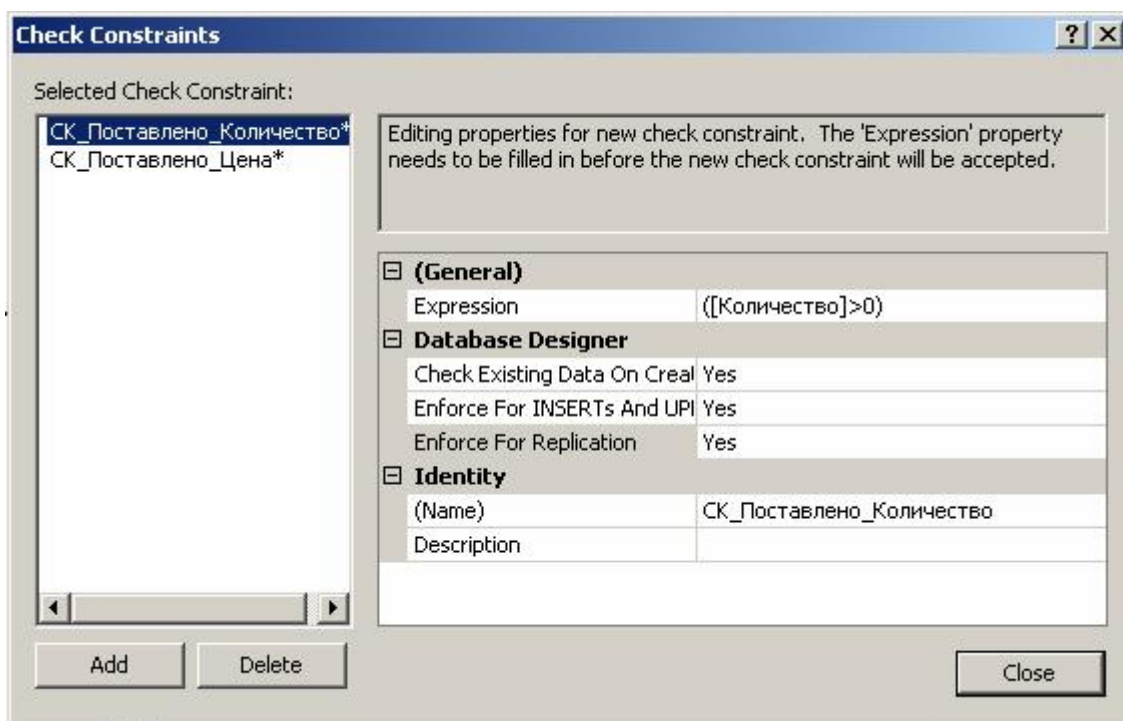


Рисунок 1.17

15. Аналогично можно сформировать контрольное выражение для поля «Цена». В этом случае выражение (Expression) будет иметь вид: ([Цена]>0), а имя (Name): СК_Поставлено_Цена. После внесения этих изменений диаграмму можно закрыть и сохранить.
16. После закрытия диаграммы необходимо проанализировать структурные изменения, сделанные в таблицах (появление новых ключей и т.д.). Для этого следует проанализировать объекты каждой таблиц, последовательно открывая таблиц в списке таблиц.

II. Ввод данных в таблицы базы данных

1. Для ввода информации в таблицу нужно выбрать таблицу в списке таблиц, щелкнув по ней правой кнопкой мыши, и в появившемся меню выбрать пункт Open Table. В результате таблица будет выведена на экран в виде, позволяющем вводить новые данные или корректировать введенные ранее.
2. Используя интерактивные средства SQL Server Management Studio необходимо ввести в таблицы базы данных информацию, приведенную на рисунках 1.18 – 1.22.

	КодПоставщика	Адрес	Примечание
▶	1	г. Харьков, пр. Ленина, 55, к.108	тел. 32-18-44
	2	г. Киев, пр. Победы, 154, к. 3	
	3	г. Харьков, ул. Пушкинская, 77	тел.33-33-44, 12-34-56, факс 22-12-33
	4	г. Одесса, ул. Дерибасовская, 75	
	5	г. Полтава, ул. Ленина, 15, кв. 43	
*	NULL	NULL	NULL

Рисунок 1.18 – Данные, введенные в таблицу «Поставщики»

	КодПоставщика	Фамилия	Имя	Отчество	НомерСвидетельства
▶	1	Петров	Павел	Петрович	12345678
	3	Иванов	Илья	Ильич	00123987
	5	Сидоров	Сергей	Степанович	09876541
*	NULL	NULL	NULL	NULL	NULL

Рисунок 1.19 – Данные, введенные в таблицу «ФизическиеЛица»

	КодПоставщика	Название	НалоговыйНомер	НомерСвидетельстваНДС
▶	2	ООО "Интерфрут"	00123987	19848521
	4	ЗАО "Транссервис"	29345678	25912578
*	NULL	NULL	NULL	NULL

Рисунок 1.20 – Данные, введенные в таблицу «ЮридическиеЛица»

Внимание! При вводе данных в таблицу «Договоры» следует учитывать, что номер каждого договора определяется автоматически.

	НомерДоговора	ДатаДоговора	КодПоставщика	Комментарий
►	1	01.09.1999 0:00:00	1	Основание - накладная № 34 от 30/08/99
	2	10.09.1999 0:00:00	1	Основание – счет-фактура № 08-78 от 28/08/99
	3	10.09.1999 0:00:00	3	Основание – счет-фактура № 08-178 от 29/08/99
	4	23.09.1999 0:00:00	3	Основание – заказ № 56 от 28/08/99
	5	24.09.1999 0:00:00	2	Основание – накладная № 74 от 11/09/99
	6	01.10.1999 0:00:00	1	Основание – счет-фактура № 09-12 от 28/09/99
	7	02.10.1999 0:00:00	2	Основание – накладная № 85 от 21/09/99
*	NULL	NULL	NULL	NULL

Рисунок 1.21 – Данные, введенные в таблицу «Договоры»

	НомерДоговора	Товар	Количество	Цена
►	1	Видеомагнитофон	12	722,33
	1	Компьютер	24	1554,22
	1	Магнитофон	25	655,12
	1	Стереосистема	12	220,45
	1	Телевизор	10	1253,45
	2	Видеомагнитофон	8	450,67
	2	Компьютер	43	1453,18
	2	Магнитофон	5	455,14
	2	Стереосистема	11	511,43
	3	Магнитофон	11	544,00
	3	Монитор	85	545,32
	3	Телевизор	52	899,99
	4	Магнитофон	22	323,19
	4	Принтер	41	350,77
	4	Стереосистема	27	330,55
	4	Телевизор	56	990,56
	5	Видеомагнитофон	17	850,12
	5	Магнитофон	33	585,67
	5	Монитор	44	590,23
	5	Телевизор	14	860,33
	6	Компьютер	32	1850,24
	6	Монитор	51	520,95
	6	Телевизор	34	810,15
	7	Компьютер	15	1234,56
	7	Монитор	22	389,75
	7	Телевизор	62	900,58
*	NULL	NULL	NULL	NULL

Рисунок 1.22 – Данные, введенные в таблицу «Поставлено»

III. Отключение и подключение базы данных

В процессе работы с базой данных может возникнуть необходимость копирования файлов базы данных с целью создания резервной копии и т.д. В СУБД Microsoft SQL Server 2005 существует несколько способов создания

копий баз данных. Одним из простейших способов является отключение и подключение базы данных. Для отключения и подключения базы данных нужно выполнить следующую последовательность действий

1. Выбрать отключаемую базу данных (в данном случае – ранее созданную базу данных delivery)
2. Щелкнуть по базе данных правой кнопкой мыши и в появившемся меню выбрать пункт Tasks. Этому пункту соответствует подменю, в котором нужно выбрать пункт Detach.... Затем в появившемся окне Detach Database нужно нажать кнопку ОК. В результате отключенная база данных исчезнет из списка баз данных, а файлы базы данных станут доступными для выполнения файловых манипуляций.
3. Отключенную базу данных можно вновь подключить. Для этого в окне Object Explorer нужно щелкнуть правой кнопкой мыши по пункту Databases и в появившемся меню выбрать пункт Attach.... Затем в появившемся окне Attach Databases нужно нажать кнопку Add и выбрать подключаемую базу данных, указав местоположение ее файлов. После этого нужно нажать кнопку ОК. В результате база данных появится в списке баз данных
4. Проверить возможность работы с базой данных (т.е. наличие объектов вновь подключенной базы данных, наличие данных в таблицах и т.д.).

IV. Сохранение результатов работы

Отключить базу данных.

Сохранить файлы базы данных delivery.mdf, delivery_log.ldf

Требования к отчету:

- 1) кратко описать основные этапы выполнения задания
- 2) изобразить структуру созданной базы данных и отношения между таблицами
- 3) описать введенную в базу данных информацию.

Лабораторная работа 2

Создание объектов базы данных и ввод информации в базу данных на основе использования средств языка SQL

Действия, рассмотренные в лабораторной работе 1, могут быть выполнены не только в интерактивном режиме, но и на основе использования языковых средств DDL и DML языка SQL. Для этого необходимо создать новую базу данных (например, с именем dlvr). Последовательность действий при создании базы данных аналогична действиям в лабораторной работе 1. Для размещения файлов базы данных можно указать тот же каталог.

ВЫПОЛНЕНИЕ РАБОТЫ

I. Использование средств DDL для создание объектов базы данных

В среде SQL Server Management Studio с базой данных можно, используя непосредственно операторы языка SQL. Для этого необходимо создать один или несколько запросов. Каждый запрос может содержать произвольное количество операторов языка SQL. Рассмотрим последовательность действий при создании запроса, с помощью которого будут созданы таблицы базы данных и связи между ними.

1. На панели инструментов нажать кнопку New Query
2. Ввести текст запроса, приведенный на рисунке 2.1.

```
USE dlvr
CREATE TABLE Поставщики (КодПоставщика int PRIMARY KEY,
                           Адрес text NOT NULL,
                           Примечание text)

CREATE TABLE ФизическиеЛица (КодПоставщика int PRIMARY KEY,
                              Фамилия char(20) NOT NULL,
                              Имя char(20) NOT NULL,
                              Отчество char(20) NOT NULL,
                              НомерСвидетельства char(10)
                              FOREIGN KEY (КодПоставщика) REFERENCES Поставщики(КодПоставщика))

CREATE TABLE ЮридическиеЛица (КодПоставщика int PRIMARY KEY,
                                Название char(20) NOT NULL,
                                НалоговыйНомер char(20),
                                НомерСвидетельстваНДС char(10)
                                FOREIGN KEY (КодПоставщика) REFERENCES Поставщики(КодПоставщика))

CREATE TABLE Договоры (НомерДоговора int IDENTITY (1,1) PRIMARY KEY,
                        ДатаДоговора datetime,
                        КодПоставщика int NOT NULL,
                        Комментарий text
                        FOREIGN KEY (КодПоставщика) REFERENCES Поставщики(КодПоставщика))

CREATE TABLE Поставлено (НомерДоговора int,
                           Товар char(20),
                           Количество decimal(4,0) NOT NULL CHECK (Количество>0),
                           Цена decimal(8,2) NOT NULL CHECK (Цена>0)
                           FOREIGN KEY (НомерДоговора) REFERENCES Договоры(НомерДоговора)
                           PRIMARY KEY (НомерДоговора, Товар))
```

Рисунок 2.1

3. Выполнить запрос. Для этого на панели инструментов нужно нажать кнопку Execute. В том случае, если текст запроса не содержит ошибок, на экране появится окно Messages с сообщением Command(s) completed successfully. В противном случае будет выведена информация об имеющихся в тексте запроса ошибках.
4. В случае успешного выполнения запроса далее следует проверить наличие объектов базы данных. В том случае, если список таблиц сразу не отображается, можно щелкнуть правой кнопкой мыши по имени базы данных и в появившемся меню выбрать пункт Refresh.
5. Созданный запрос можно закрыть и сохранить с произвольным именем (например, SQLQuery_create_tables.sql)

С помощью операторов DDL языка SQL можно не только создавать объекты базы данных, но и изменять структуру ранее созданных объектов. Предположим, что в таблице «Поставлено» размер поля «Количество» может не соответствовать реальным значениям хранимых данных. В связи с этим размер поля необходимо увеличить. Это можно сделать с помощью следующего запроса (рисунок 2.2)

```
use dlvr  
  
ALTER TABLE Поставлено ALTER COLUMN Количество decimal (5,0) NOT NULL
```

Рисунок 2.2

Последовательность действий при создании и выполнении запроса аналогична последовательности действий, рассмотренных выше. Созданный запрос можно закрыть и сохранить с произвольным именем (например, SQLQuery_alter_tables.sql)

II. Использование средств DML для ввода информации в таблицы базы данных

Запросы могут содержать не только операторы DDL, но и операторы DML. Это позволяет реализовать основные операции манипулирования данными. Рассмотрим последовательность действий при создании запроса, с помощью которого в таблицы созданной базы данных будет введена информация.

1. На панели инструментов нажать кнопку New Query
2. Ввести текст запроса, приведенный на рисунках 2.3 – 2.5.

```

USE dlvr

INSERT INTO Поставщики (КодПоставщика, Адрес, Примечание)
VALUES (1, 'г. Харьков, пр. Ленина, 55, к.108', 'тел. 32-18-44');
INSERT INTO Поставщики (КодПоставщика, Адрес, Примечание)
VALUES (2, 'г. Киев, пр. Победы, 154, к. 3', '');
INSERT INTO Поставщики (КодПоставщика, Адрес, Примечание)
VALUES (3, 'г. Харьков, ул. Пушкинская, 77', 'тел.33-33-44, 12-34-56, факс 22-12-33');
INSERT INTO Поставщики (КодПоставщика, Адрес, Примечание)
VALUES (4, 'г. Одесса, ул. Дерибасовская, 75', '');
INSERT INTO Поставщики (КодПоставщика, Адрес, Примечание)
VALUES (5, 'г. Полтава, ул. Ленина, 15, кв. 43', '');

INSERT INTO ФизическиеЛица
VALUES (3, 'Иванов', 'Илья', 'Ильич', '00123987');
INSERT INTO ФизическиеЛица
VALUES (1, 'Петров', 'Павел', 'Петрович', '12345678');
INSERT INTO ФизическиеЛица
VALUES (5, 'Сидоров', 'Сергей', 'Степанович', '09876541');

INSERT INTO ЮридическиеЛица
VALUES (2, 'ООО "Интерфрут"', '00123987', '19848521');
INSERT INTO ЮридическиеЛица
VALUES (4, 'ЗАО "Транссервис"', '29345678', '25912578');

```

Рисунок 2.3

```

INSERT INTO Договоры (ДатаДоговора, КодПоставщика, Комментарий)
VALUES ('19990901', 1, 'Основание - накладная № 34 от 30/08/99');
INSERT INTO Договоры (ДатаДоговора, КодПоставщика, Комментарий)
VALUES ('1999/09/10', 1, 'Основание - счет-фактура № 08-78 от 28/08/99');
INSERT INTO Договоры (ДатаДоговора, КодПоставщика, Комментарий)
VALUES ('19990910', 3, 'Основание - счет-фактура № 08-178 от 29/08/99');
INSERT INTO Договоры (ДатаДоговора, КодПоставщика, Комментарий)
VALUES ('19990923', 3, 'Основание - заказ № 56 от 28/08/99');
INSERT INTO Договоры (ДатаДоговора, КодПоставщика, Комментарий)
VALUES ('19990924', 2, 'Основание - накладная № 74 от 11/09/99');
INSERT INTO Договоры (ДатаДоговора, КодПоставщика, Комментарий)
VALUES ('1999/10/01', 1, 'Основание - счет-фактура № 09-12 от 28/09/99');
INSERT INTO Договоры (ДатаДоговора, КодПоставщика, Комментарий)
VALUES ('19991002', 2, 'Основание - накладная № 85 от 21/09/99');

```

Рисунок 2.3

```

INSERT INTO Поставлено VALUES (1, 'Телевизор', 10, 1253.45);
INSERT INTO Поставлено VALUES (1, 'Магнитофон', 25, 655.12);
INSERT INTO Поставлено VALUES (1, 'Видеомагнитофон', 12, 722.33);
INSERT INTO Поставлено VALUES (2, 'Стереосистема', 11, 511.43);
INSERT INTO Поставлено VALUES (2, 'Магнитофон', 5, 455.14);
INSERT INTO Поставлено VALUES (2, 'Видеомагнитофон', 8, 450.67);
INSERT INTO Поставлено VALUES (1, 'Стереосистема', 12, 220.45);
INSERT INTO Поставлено VALUES (1, 'Компьютер', 24, 1554.22);
INSERT INTO Поставлено VALUES (2, 'Компьютер', 43, 1453.18);
INSERT INTO Поставлено VALUES (3, 'Телевизор', 52, 899.99);
INSERT INTO Поставлено VALUES (3, 'Магнитофон', 11, 544.00);
INSERT INTO Поставлено VALUES (3, 'Монитор', 85, 545.32);
INSERT INTO Поставлено VALUES (4, 'Телевизор', 56, 990.56);
INSERT INTO Поставлено VALUES (4, 'Магнитофон', 22, 323.19);
INSERT INTO Поставлено VALUES (4, 'Принтер', 41, 350.77);
INSERT INTO Поставлено VALUES (5, 'Телевизор', 14, 860.33);
INSERT INTO Поставлено VALUES (5, 'Магнитофон', 33, 585.67);
INSERT INTO Поставлено VALUES (5, 'Видеомагнитофон', 17, 850.12);
INSERT INTO Поставлено VALUES (4, 'Стереосистема', 27, 330.55);
INSERT INTO Поставлено VALUES (5, 'Монитор', 44, 590.23);
INSERT INTO Поставлено VALUES (6, 'Телевизор', 34, 810.15);
INSERT INTO Поставлено VALUES (6, 'Компьютер', 32, 1850.24);
INSERT INTO Поставлено VALUES (6, 'Монитор', 51, 520.95);
INSERT INTO Поставлено VALUES (7, 'Телевизор', 62, 900.58);
INSERT INTO Поставлено VALUES (7, 'Компьютер', 15, 1234.56);
INSERT INTO Поставлено VALUES (7, 'Монитор', 22, 389.75);

```

Рисунок 2.3

3. Выполнить запрос. Для этого на панели инструментов нужно нажать кнопку Execute. В том случае, если текст запроса не содержит ошибок, на экране появится окно Messages с сообщениями типа (1 row(s) affected). В противном случае будет выведена информация об имеющихся в тексте запроса ошибках.
4. В случае успешного выполнения запроса далее следует проверить наличие информации в таблицах базы данных. Для этого нужно выбрать таблицу, щелкнув по ней правой кнопкой мыши, и в появившемся меню выбрать пункт Open Table.
5. Созданный запрос можно закрыть и сохранить с произвольным именем (например, SQLQuery_insert.sql)

III. Сопоставление созданных баз данных

В результате выполнения лабораторных работ 1 и 2 были созданы практически одинаковые базы данных. Тем не менее, в этих базах данных могут быть определенные отличия. Необходимо проанализировать объекты баз данных, выявить отличия (если таковые имеются) и установить причину их появления. Также необходимо создать в новой базе данных диаграмму. При создании диаграммы следует обратить внимание на то, что связи между таблицами в

диаграмме появляются автоматически при их добавлении в диаграмму. Сопоставить диаграмму с диаграммой, созданной в базе данных при выполнении лабораторной работы 1.

IV. Сохранение результатов работы

Отключить базу данных.

Сохранить файлы базы данных – dlvr.mdf, dlvr_log.ldf

Сохранить файлы, содержащие тексты запросов – SQLQuery_create_tables.sql, SQLQuery_alter_tables.sql, SQLQuery_insert.sql

Требования к отчету:

- 1) кратко описать основные этапы выполнения задания;
- 2) привести текст запросов, реализованных при выполнении лабораторной работы, описать назначение операторов, особенности их структуры и использования;
- 3) изобразить структуру созданной базы данных и отношения между таблицами;
- 4) описать расхождения между базами данных, созданными результате выполнения лабораторных работ 1 и 2 и проанализировать причины появления этих расхождений

Лабораторная работа 3

Использование оператора SELECT – SQL для обработки данных

Для выполнения работы необходимо подключить базу данных, которая была создана и заполнена данными в процессе выполнения лабораторной работы 1. Основной целью данной работы является изучение особенностей использования оператора SELECT-SQL при разработке запросов в среде SQL Server Management Studio, а также рассмотрение некоторых особенностей реализации оператора SELECT-SQL в языке Transact-SQL (T-SQL).

Рассмотрим последовательность действий по созданию и выполнению запроса, позволяющего обрабатывать данные с помощью оператора SELECT-SQL с помощью утилиты ISQL на примере запроса 1.

Запрос 1

Условие

Вывести на экран список товаров, поставленных поставщиком 1 (ЧП Иванов И.И.) по договору 1.

Создание и выполнение запроса.

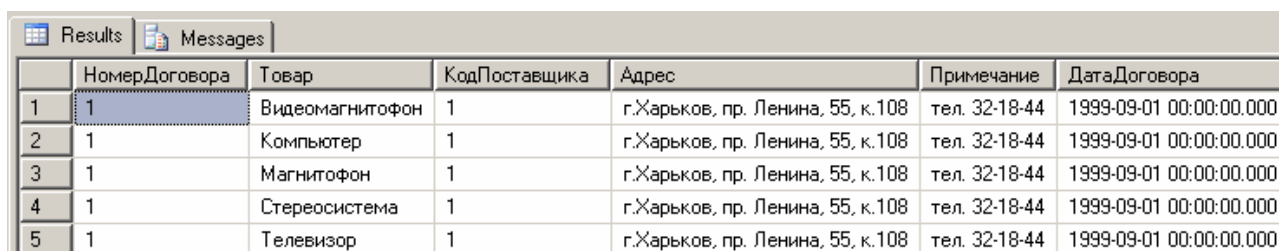
1. На панели инструментов нажать кнопку New Query
2. Ввести текст запроса, приведенный на рисунке 3.1

```
USE delivery
```

```
SELECT Поставлено.НомерДоговора, Поставлено.Товар, Поставщики.*, Договоры.ДатаДоговора  
FROM Поставлено, Договоры, Поставщики  
WHERE Договоры.НомерДоговора = Поставлено.НомерДоговора  
AND Поставщики.КодПоставщика = Договоры.КодПоставщика AND (Договоры.НомерДоговора = 1  
AND Договоры.КодПоставщика = 1)
```

Рисунок 3.1

3. Нажать кнопку «Execute». В том случае, если в тексте запроса нет ошибок, будет выведен результат запроса. Этот результат может иметь вид (рисунок 3.2).



	НомерДоговора	Товар	КодПоставщика	Адрес	Примечание	ДатаДоговора
1	1	Видеомагнитофон	1	г.Харьков, пр. Ленина, 55, к.108	тел. 32-18-44	1999-09-01 00:00:00.000
2	1	Компьютер	1	г.Харьков, пр. Ленина, 55, к.108	тел. 32-18-44	1999-09-01 00:00:00.000
3	1	Магнитофон	1	г.Харьков, пр. Ленина, 55, к.108	тел. 32-18-44	1999-09-01 00:00:00.000
4	1	Стереосистема	1	г.Харьков, пр. Ленина, 55, к.108	тел. 32-18-44	1999-09-01 00:00:00.000
5	1	Телевизор	1	г.Харьков, пр. Ленина, 55, к.108	тел. 32-18-44	1999-09-01 00:00:00.000

Рисунок 3.2

4. Текст запроса можно сохранить в виде файла (например, SQLQuery01_1.sql). В том случае, если в дальнейшем этот запрос нужно будет выполнить повторно или изменить, можно открыть файл запроса. Для этого в главном меню нужно выбрать пункт File, а затем в вертикальном меню выбрать пункт Open, подпункт File и выбрать соответствующий файл.

Как видно из текста запроса, этот запрос является многотабличным, причем таблицы соединяются на основе использования естественного соединения. В случае использования открытого соединения этот запрос имел бы вид (рисунок 3.3). Этот запрос также необходимо создать и выполнить для проверки работоспособности, а затем сохранить в файле с именем SQLQuery01_2.sql

```
USE delivery|

SELECT Поставлено.НомерДоговора, Поставлено.Товар, Поставщики.*, Договоры.ДатаДоговора
FROM (Поставщики INNER JOIN Договоры ON Поставщики.КодПоставщика = Договоры.КодПоставщика)
     INNER JOIN Поставлено ON Договоры.НомерДоговора = Поставлено.НомерДоговора
WHERE Договоры.НомерДоговора = 1 AND Договоры.КодПоставщика = 1
```

Рисунок 3.3

Создание и выполнение остальных запросов выполняется аналогично. Поэтому далее будет приведено назначение каждого запроса и текст.

Внимание! Все рассматриваемые запросы должны быть результативными (т.е. в результате выполнения запроса должны быть выведены одна или несколько записей). Отсутствие результата запроса является признаком ошибок при построении запроса, несоответствия запроса имеющимся данным и т.д. Такой запрос нуждается в анализе и проверке.

Запрос 2

Вывести на экран список товаров, поставленных поставщиком 1 (ЧП Иванов И.И.) в период с 05/09/1999 по 12/09/1999.

Текст запроса приведен на рисунке 3.4

```
USE delivery

SELECT Договоры.НомерДоговора, Договоры.ДатаДоговора, Поставлено.Товар,
       Поставлено.Цена, Поставщики.*
FROM (Поставщики INNER JOIN Договоры ON Поставщики.КодПоставщика = Договоры.КодПоставщика)
     INNER JOIN Поставлено ON Договоры.НомерДоговора = Поставлено.НомерДоговора
WHERE Договоры.ДатаДоговора BETWEEN '19990905' and '19990912' AND
       Поставщики.КодПоставщика = 1
```

Рисунок 3.4

Запрос можно сохранить в файле с именем SQLQuery02.sql

Запрос 3

Вывести на экран список товаров, поставленных в 9 месяце 1999 года с выводом наименования поставщика и даты поставки.

Текст запроса приведен на рисунке 3.5

```
USE delivery|

SELECT Договоры.НомерДоговора, Договоры.ДатаДоговора, Поставлено.Товар,
       Поставлено.Цена, Поставщики.*
FROM (Поставщики INNER JOIN Договоры ON Поставщики.КодПоставщика = Договоры.КодПоставщика)
     INNER JOIN Поставлено ON Договоры.НомерДоговора = Поставлено.НомерДоговора
WHERE MONTH(Договоры.ДатаДоговора)=9 AND YEAR(Договоры.ДатаДоговора)=1999
```

Рисунок 3.5

Запрос можно сохранить в файле с именем SQLQuery03.sql

Запрос 4

Вывести на экран список договоров (номер, дата, название) и общую сумму по каждому договору (размер партии умножить на цену за штуку и просуммировать по договору). Список должен быть отсортирован в порядке возрастания номеров договоров.

Текст запроса приведен на рисунке 3.6

```
USE delivery

SELECT Договоры.НомерДоговора, Договоры.ДатаДоговора, Договоры.КодПоставщика,
       SUM(Цена*Количество) AS Сумма
FROM Договоры INNER JOIN Поставлено
     ON Договоры.НомерДоговора = Поставлено.НомерДоговора
GROUP BY Договоры.НомерДоговора, Договоры.ДатаДоговора, Договоры.КодПоставщика
ORDER BY Договоры.НомерДоговора
```

Рисунок 3.6

Запрос можно сохранить в файле с именем SQLQuery04.sql

Запрос 5

Вывести на экран список договоров (номер, дата, название) и общую сумму по каждому договору (размер партии умножить на цену за штуку и просуммировать по договору). Список должен быть отсортирован в порядке возрастания общих сумм по каждому договору. После этого на список должно быть наложено условие фильтрации, состоящее в исключении из результата запроса записей, для которых номер договора больше 3.

Текст запроса приведен на рисунке 3.7

```
USE delivery|

SELECT Договоры.НомерДоговора, Договоры.ДатаДоговора, Договоры.КодПоставщика,
       SUM(Цена*Количество) AS Сумма
FROM   Договоры INNER JOIN Поставлено
       ON Договоры.НомерДоговора = Поставлено.НомерДоговора
WHERE  Договоры.НомерДоговора > 3
GROUP BY Договоры.НомерДоговора, Договоры.ДатаДоговора, Договоры.КодПоставщика
ORDER BY Договоры.НомерДоговора
```

Рисунок 3.7

Запрос можно сохранить в файле с именем SQLQuery05.sql

Запрос 6

Вывести на экран сведения о наибольшей по размеру партии товара во всех договорах с указанием поставщика, а также номера и даты договора.

Текст запроса приведен на рисунке 3.8

```
USE delivery

SELECT Договоры.НомерДоговора, Договоры.ДатаДоговора,
       Договоры.Комментарий, Поставщики.*, Поставлено.Цена
FROM   Договоры, Поставлено, Поставщики
WHERE  Договоры.НомерДоговора = Поставлено.НомерДоговора AND
       Договоры.КодПоставщика = Поставщики.КодПоставщика AND
       Поставлено.Цена = (SELECT MAX(Поставлено.Цена) FROM Поставлено)
```

Рисунок 3.8

Запрос можно сохранить в файле с именем SQLQuery06.sql

Запрос 7

Вывести на экран список поставщиков (наименование и код), с которыми не было заключено ни одного договора.

Текст запроса приведен на рисунке 3.9

```
USE delivery

SELECT * FROM Поставщики
WHERE КодПоставщика NOT IN (SELECT КодПоставщика FROM Договоры)
```

Рисунок 3.9

Запрос можно сохранить в файле с именем SQLQuery07.sql

Запрос 8

Вывести на экран список наименований поставленных товаров с указанием средней цены поставки за единицу (вне зависимости от поставщика).

Текст запроса приведен на рисунке 3.10

```
USE delivery

SELECT Товар, AVG(Цена) AS СредняяЦена
FROM Поставлено
GROUP BY Товар
```

Рисунок 3.10

Запрос можно сохранить в файле с именем SQLQuery08.sql

Запрос 9

Вывести на экран список товаров (наименование, количество и цена, поставщик), для которых цена за единицу больше средней.

Текст запроса приведен на рисунке 3.11

```
USE delivery

SELECT Товар, Количество, Цена, Поставщики.*
FROM (Поставщики INNER JOIN Договоры ON Поставщики.КодПоставщика = Договоры.КодПоставщика)
INNER JOIN Поставлено ON Договоры.НомерДоговора = Поставлено.НомерДоговора
WHERE Цена > (SELECT AVG(Цена) FROM Поставлено)
```

Рисунок 3.11

Запрос можно сохранить в файле с именем SQLQuery09.sql

Запрос 10

Вывести на экран сведения о пяти самых дорогих товарах (наименование, цена за единицу, поставщик).

Текст запроса приведен на рисунке 3.12

```
USE delivery

SELECT TOP 5 Товар, Цена, Поставщики.*
FROM (Поставщики INNER JOIN Договоры ON Поставщики.КодПоставщика = Договоры.КодПоставщика)
INNER JOIN Поставлено ON Договоры.НомерДоговора = Поставлено.НомерДоговора
ORDER BY Цена DESC
```

Рисунок 3.12

Запрос можно сохранить в файле с именем SQLQuery10.sql

Запрос 11

Сформировать список поставщиков с указанием кода, адреса и данных поставщика. При формировании данных поставщика для поставщиков – физических лиц вывести фамилию и инициалы, для поставщиков – юридических лиц – название.

Текст запроса приведен на рисунке 3.13

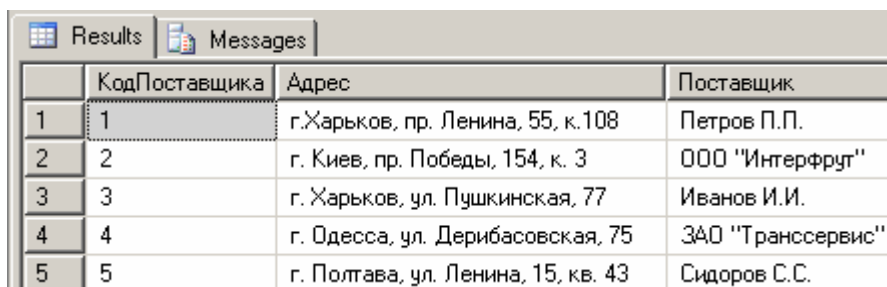
```
use delivery

select Поставщики.КодПоставщика, Поставщики.Адрес,
isnull(ЮридическиеЛица.Название, rtrim(ФизическиеЛица.Фамилия) + ' ' +
substring(ФизическиеЛица.Имя, 1, 1) + '.' +
substring(ФизическиеЛица.Отчество, 1, 1) + '.') as Поставщик
from (Поставщики left join ФизическиеЛица
on Поставщики.КодПоставщика=ФизическиеЛица.КодПоставщика)
left join ЮридическиеЛица
on Поставщики.КодПоставщика=ЮридическиеЛица.КодПоставщика
```

Рисунок 3.13

Запрос можно сохранить в файле с именем SQLQuery11.sql.

Результат запроса может иметь вид, приведенный на рисунке 3.14.



	КодПоставщика	Адрес	Поставщик
1	1	г.Харьков, пр. Ленина, 55, к.108	Петров П.П.
2	2	г. Киев, пр. Победы, 154, к. 3	ООО "Интерфрут"
3	3	г. Харьков, ул. Пушкинская, 77	Иванов И.И.
4	4	г. Одесса, ул. Дерибасовская, 75	ЗАО "Транссервис"
5	5	г. Полтава, ул. Ленина, 15, кв. 43	Сидоров С.С.

Рисунок 3.14

Запрос 12

Сформировать список договоров (с указанием номера, даты поставки и данных о поставщике), общее количество поставленных товаров и общую сумму по каждому договору. Для поставщиков – физических лиц вывести фамилию и инициалы, для поставщиков – юридических лиц – номер свидетельства плательщика НДС. В результат запроса должны быть включены только те договоры, на основании которых товары действительно поставлялись (т.е. в результат запроса не должны попасть так называемые «пустые» договоры)

Текст запроса приведен на рисунке 3.15

Запрос можно сохранить в файле с именем SQLQuery12.sql.

```

use delivery

select Договоры.НомерДоговора, Договоры.ДатаДоговора,
       isnull(ЮридическиеЛица.Название, rtrim(ФизическиеЛица.Фамилия) + ' ' +
       substring(ФизическиеЛица.Имя, 1, 1) + '.' +
       substring(ФизическиеЛица.Отчество, 1, 1) + '.') as Поставщик,
       Sum(Поставлено.Количество) AS ОбъемПоставки,
       Sum(Количество*Цена) AS СуммаПоставки
from ((Поставщики LEFT JOIN ФизическиеЛица
      ON Поставщики.КодПоставщика=ФизическиеЛица.КодПоставщика)
LEFT JOIN ЮридическиеЛица
      ON Поставщики.КодПоставщика=ЮридическиеЛица.КодПоставщика)
INNER JOIN Договоры ON Договоры.КодПоставщика=Поставщики.КодПоставщика)
INNER JOIN Поставлено ON Договоры.НомерДоговора=Поставлено.НомерДоговора
group by Договоры.НомерДоговора, Договоры.ДатаДоговора,
       isnull(ЮридическиеЛица.Название, rtrim(ФизическиеЛица.Фамилия) + ' ' +
       substring(ФизическиеЛица.Имя, 1, 1) + '.' +
       substring(ФизическиеЛица.Отчество, 1, 1) + '.')
order by НомерДоговора

```

Рисунок 3.15

Запрос 13

Сформировать список товаров (с указанием номера договора и даты поставки), поставленных поставщиками 1 (ЧП Петров П.П.) и 2 (ООО «Интерфрут»).

Примечание. Данный запрос иллюстрирует особенности использования операции объединения (UNION). Нетрудно заметить, что данный запрос может быть легко реализован без использования операции объединения.

Текст запроса приведен на рисунке 3.16

```

USE delivery

SELECT Поставлено.НомерДоговора, Договоры.ДатаДоговора,
       Поставлено.Товар, Поставщики.КодПоставщика
FROM Поставлено, Договоры, Поставщики
WHERE Договоры.НомерДоговора = Поставлено.НомерДоговора
AND Поставщики.КодПоставщика = Договоры.КодПоставщика AND Договоры.КодПоставщика = 1
UNION
SELECT Поставлено.НомерДоговора, Договоры.ДатаДоговора,
       Поставлено.Товар, Поставщики.КодПоставщика
FROM Поставлено, Договоры, Поставщики
WHERE Договоры.НомерДоговора = Поставлено.НомерДоговора
AND Поставщики.КодПоставщика = Договоры.КодПоставщика AND Договоры.КодПоставщика = 2
ORDER BY КодПоставщика, НомерДоговора

```

Рисунок 3.16

Запрос можно сохранить в файле с именем SQLQuery13.sql

Запрос 14

Сформировать номенклатуру товаров (т.е. список наименований товаров), которые поставлялись только поставщиком 1 (ЧП Петров П.П.), или только поставщиком 2 (ООО «Интерфрут»), или и поставщиком 1, и поставщиком 2.

Текст запроса приведен на рисунке 3.17

```
USE delivery

SELECT DISTINCT Поставлено.Товар
FROM Поставлено, Договоры
WHERE Договоры.НомерДоговора = Поставлено.НомерДоговора AND Договоры.КодПоставщика = 1
UNION
SELECT DISTINCT Поставлено.Товар
FROM Поставлено, Договоры
WHERE Договоры.НомерДоговора = Поставлено.НомерДоговора AND Договоры.КодПоставщика = 2
ORDER BY Товар
```

Рисунок 3.17

Запрос можно сохранить в файле с именем SQLQuery14.sql

Запрос 15

Сформировать номенклатуру товаров (т.е. список наименований товаров), которые поставлялись и поставщиком 1 (ЧП Петров П.П.), и поставщиком 2 (ООО «Интерфрут»).

Примечание. Данный запрос иллюстрирует особенности использования операции пересечения (INTERSECT).

Текст запроса приведен на рисунке 3.18

```
USE delivery

SELECT DISTINCT Поставлено.Товар
FROM Поставлено, Договоры
WHERE Договоры.НомерДоговора = Поставлено.НомерДоговора AND Договоры.КодПоставщика = 1
INTERSECT
SELECT DISTINCT Поставлено.Товар
FROM Поставлено, Договоры
WHERE Договоры.НомерДоговора = Поставлено.НомерДоговора AND Договоры.КодПоставщика = 2
ORDER BY Товар
```

Рисунок 3.18

Запрос можно сохранить в файле с именем SQLQuery15.sql

Запрос 16

Сформировать номенклатуру товаров (т.е. список наименований товаров), которые поставлялись поставщиком 1 (ЧП Петров П.П.), но не поставлялись поставщиком 2 (ООО «Интерфрут»).

Примечание. Данный запрос иллюстрирует особенности использования операции разности (EXCEPT).

Текст запроса приведен на рисунке 3.19

```
USE delivery

SELECT DISTINCT Поставлено.Товар
FROM Поставлено, Договоры
WHERE Договоры.НомерДоговора = Поставлено.НомерДоговора AND Договоры.КодПоставщика = 1
EXCEPT
SELECT DISTINCT Поставлено.Товар
FROM Поставлено, Договоры
WHERE Договоры.НомерДоговора = Поставлено.НомерДоговора AND Договоры.КодПоставщика = 2
ORDER BY Товар
```

Рисунок 3.19

Запрос можно сохранить в файле с именем SQLQuery16.sql

Запрос 17

Сформировать список товаров, который должен отражать частоту поставок товаров. В список включить только товары, которые поставлялись более одного раза. Список должен быть отсортирован в порядке убывания частоты поставок.

Текст запроса приведен на рисунке 3.20

```
USE delivery

SELECT Товар, COUNT(Товар) AS ЧастотаПоставок
FROM Поставлено
GROUP BY Товар
HAVING COUNT(Товар) >1
ORDER BY COUNT(Товар) DESC
```

Рисунок 3.20

Запрос можно сохранить в файле с именем SQLQuery17.sql

Запрос 18

Сформировать данные о количественной динамике поставок товаров в течение 1999 года. Данные должны быть агрегированы по-месячно и представлены в виде таблицы, строками которой являются названия товаров, а столбцами – номера месяцев 1999 года. На пересечении строки и столбца должно отображаться количество данного товара, поставленного в данном месяце.

Примечание. Данный запрос иллюстрирует особенности создания и использования перекрестного запроса средствами языка Transact-SQL.

Текст запроса приведен на рисунке 3.21

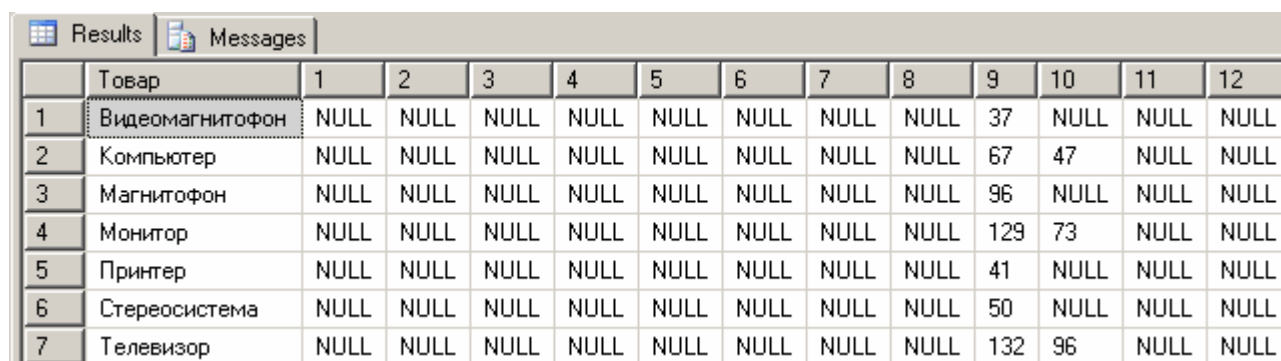
Результат запроса может иметь вид, приведенный на рисунке 3.22.

Запрос можно сохранить в файле с именем SQLQuery18_1.sql

```
USE delivery

SELECT Товар, [1],[2],[3],[4],[5],[6],[7],[8],[9],[10],[11],[12]
FROM
(
SELECT Товар, MONTH(ДатаДоговора) AS месас, Количество
FROM Договоры, Поставлено
WHERE Договоры.НомерДоговора=Поставлено.НомерДоговора AND YEAR(ДатаДоговора)=1999
) p
PIVOT
(SUM(Количество)
FOR месас IN ([1],[2],[3],[4],[5],[6],[7],[8],[9],[10],[11],[12])
) AS pvt
ORDER BY Товар
```

Рисунок 3.21



	Товар	1	2	3	4	5	6	7	8	9	10	11	12
1	Видеомагнитофон	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	37	NULL	NULL	NULL
2	Компьютер	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	67	47	NULL	NULL
3	Магнитофон	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	96	NULL	NULL	NULL
4	Монитор	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	129	73	NULL	NULL
5	Принтер	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	41	NULL	NULL	NULL
6	Стереосистема	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	50	NULL	NULL	NULL
7	Телевизор	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	132	96	NULL	NULL

Рисунок 3.22

Приведенный результат запроса может быть неудобным для восприятия (например, из-за наличия значений NULL). Этот недостаток может быть устранен, например, путем замены значений NULL на 0. Текст измененного запроса приведен на рисунке 3.23. Запрос можно сохранить в файле с именем SQLQuery18_2.sql

```
USE delivery

SELECT Товар, isnull([1],0) as [1], isnull([2],0) as [2], isnull([3],0) as [3],
isnull([4],0) as [4], isnull([5],0) as [5], isnull([6],0) as [6],
isnull([7],0) as [7], isnull([8],0) as [8], isnull([9],0) as [9],
isnull([10],0) as [10], isnull([11],0) as [11], isnull([12],0) as [12]
FROM
(
SELECT Товар, MONTH(ДатаДоговора) AS месас, Количество
FROM Договоры, Поставлено
WHERE Договоры.НомерДоговора=Поставлено.НомерДоговора AND YEAR(ДатаДоговора)=1999
) p
PIVOT
(SUM(Количество)
FOR месас IN ([1],[2],[3],[4],[5],[6],[7],[8],[9],[10],[11],[12])
) AS pvt
ORDER BY Товар
```

Рисунок 3.23

Результат запроса может иметь вид, приведенный на рисунке 3.24.

	Товар	1	2	3	4	5	6	7	8	9	10	11	12
1	Видеомагнитофон	0	0	0	0	0	0	0	0	37	0	0	0
2	Компьютер	0	0	0	0	0	0	0	0	67	47	0	0
3	Магнитофон	0	0	0	0	0	0	0	0	96	0	0	0
4	Монитор	0	0	0	0	0	0	0	0	129	73	0	0
5	Принтер	0	0	0	0	0	0	0	0	41	0	0	0
6	Стереосистема	0	0	0	0	0	0	0	0	50	0	0	0
7	Телевизор	0	0	0	0	0	0	0	0	132	96	0	0

Рисунок 3.24

Запрос 19

Сформировать список поставленных товаров. Для каждого товара в этом списке должны быть указаны следующие данные: номер договора, название товара, количество единиц, цена за единицу, дата поставки, название месяца и номер года.

Текст запроса приведен на рисунке 3.25

```
USE delivery

SELECT Поставлено.НомерДоговора, Поставлено.Товар,
       Поставлено.Количество, Поставлено.Цена,
       Договоры.ДатаДоговора,
       DATENAME(month, ДатаДоговора) AS Месяц,
       YEAR(ДатаДоговора) AS Год
FROM Поставлено, Договоры
WHERE Договоры.НомерДоговора = Поставлено.НомерДоговора
```

Рисунок 3.25

Результат запроса (фрагмент) может иметь вид, приведенный на рисунке 3.26. Запрос можно сохранить в файле с именем SQLQuery19_1.sql

	НомерДоговора	Товар	Количество	Цена	ДатаДоговора	Месяц	Год
16	4	Телевизор	56	990.56	1999-09-23 00:00:00.000	September	1999
17	5	Видеомагнитофон	17	850.12	1999-09-24 00:00:00.000	September	1999
18	5	Магнитофон	33	585.67	1999-09-24 00:00:00.000	September	1999
19	5	Монитор	44	590.23	1999-09-24 00:00:00.000	September	1999
20	5	Телевизор	14	860.33	1999-09-24 00:00:00.000	September	1999
21	6	Компьютер	32	1850.24	1999-10-01 00:00:00.000	October	1999
22	6	Монитор	51	520.95	1999-10-01 00:00:00.000	October	1999
23	6	Телевизор	34	810.15	1999-10-01 00:00:00.000	October	1999
24	7	Компьютер	15	1234.56	1999-10-02 00:00:00.000	October	1999
25	7	Монитор	22	389.75	1999-10-02 00:00:00.000	October	1999
26	7	Телевизор	62	900.58	1999-10-02 00:00:00.000	October	1999

Рисунок 3.26

Как видно из результата запроса, формально требование включения в результат запроса наименования месяца выполнено путем использования встроенной функции DATENAME(). Однако такие названия месяцев не всегда удобны для восприятия. Может возникнуть требование их замены на русскоязычные, украиноязычные и т.п. Эту проблему можно решить путем разработки пользовательской функции, конвертирующей названия месяцев. Этот подход несколько более трудоемкий. Другим вариантом решения проблемы может быть использование в запросе функции CASE языка Transact-SQL. Текст такого запроса приведен на рисунке 3.27. Результат запроса (фрагмент) может иметь вид, приведенный на рисунке 3.28. Запрос можно сохранить в файле с именем SQLQuery19_2.sql

```
USE delivery|

SELECT Поставлено.НомерДоговора, Поставлено.Товар,
       Поставлено.Количество, Поставлено.Цена,
       Договоры.ДатаДоговора,
       Месяц = CASE MONTH(ДатаДоговора)
                 WHEN 1 THEN 'январь'
                 WHEN 2 THEN 'февраль'
                 WHEN 3 THEN 'март'
                 WHEN 4 THEN 'апрель'
                 WHEN 5 THEN 'май'
                 WHEN 6 THEN 'июнь'
                 WHEN 7 THEN 'июль'
                 WHEN 8 THEN 'август'
                 WHEN 9 THEN 'сентябрь'
                 WHEN 10 THEN 'октябрь'
                 WHEN 11 THEN 'ноябрь'
                 WHEN 12 THEN 'декабрь'
                 ELSE '?????????'
               END,
       YEAR(ДатаДоговора) AS Год
FROM Поставлено, Договоры
WHERE Договоры.НомерДоговора = Поставлено.НомерДоговора
```

Рисунок 3.27

	НомерДогово...	Товар	Количество	Цена	ДатаДоговора	Месяц	Год
16	4	Телевизор	56	990.56	1999-09-23 00:00:00.000	сентябрь	1999
17	5	Видеомагнитофон	17	850.12	1999-09-24 00:00:00.000	сентябрь	1999
18	5	Магнитофон	33	585.67	1999-09-24 00:00:00.000	сентябрь	1999
19	5	Монитор	44	590.23	1999-09-24 00:00:00.000	сентябрь	1999
20	5	Телевизор	14	860.33	1999-09-24 00:00:00.000	сентябрь	1999
21	6	Компьютер	32	1850.24	1999-10-01 00:00:00.000	октябрь	1999
22	6	Монитор	51	520.95	1999-10-01 00:00:00.000	октябрь	1999
23	6	Телевизор	34	810.15	1999-10-01 00:00:00.000	октябрь	1999
24	7	Компьютер	15	1234.56	1999-10-02 00:00:00.000	октябрь	1999
25	7	Монитор	22	389.75	1999-10-02 00:00:00.000	октябрь	1999
26	7	Телевизор	62	900.58	1999-10-02 00:00:00.000	октябрь	1999

Рисунок 3.28

Сохранение результатов работы

Сохранить файлы запросов:

SQLQuery01_1.sql; SQLQuery01_2.sql;

SQLQuery02.sql;

SQLQuery03.sql;

SQLQuery04.sql;

SQLQuery05.sql;

SQLQuery06.sql;

SQLQuery07.sql;

SQLQuery08.sql;

SQLQuery09.sql;

SQLQuery10.sql;

SQLQuery11.sql;

SQLQuery12.sql;

SQLQuery13.sql;

SQLQuery14.sql;

SQLQuery15.sql;

SQLQuery16.sql;

SQLQuery17.sql;

SQLQuery18_1.sql; SQLQuery18_2.sql;

SQLQuery19_1.sql; SQLQuery19_2.sql

Требования к отчету:

- 1) кратко описать основные этапы выполнения работы;
- 2) для каждого из реализованных запросов привести условие запроса, текст запроса и результат выполнения запроса (в виде таблицы, рисунка, экранной формы и т.п.).

Лабораторная работа 4

Создание и использование программных объектов базы данных

Как и все СУБД, поддерживающие технологию "клиент-сервер", Microsoft SQL Server 2005 наряду с данными хранит в базах данных программные объекты. Такими объектами являются: хранимые процедуры, хранимые функции и триггеры. Программные объекты могут использоваться большим количеством приложений. Такие объекты позволяют повысить эффективность функционирования приложений, взаимодействующих с базами данных, обеспечить высокую степень защиты последних и унифицировать способы обращения к данным из приложений.

Хранимые процедуры - это подпрограммы на языке SQL, хранящиеся в базах данных и представляющие собой один из видов их общих ресурсов. Тело любой хранимой процедуры представляет последовательность SQL-операторов, например таких, как выборка данных, их модификация, удаление данных, операторы цикла, условные операторы и ряд других. Процедуры вызываются и могут использовать как входные параметры (передающие значения в процедуру), так и выходные параметры (возвращающие результаты процедуры вызывающему программному объекту). Процедуры могут вызываться из процедур, функций и других типов программных объектов.

Хранимые процедуры создаются оператором **CREATE PROCEDURE**. Модификация тела хранимой процедуры осуществляется оператором **ALTER PROCEDURE**. Эти операторы могут использовать:

- пользователи, которым разрешено создавать объекты базы данных;
- администратор базы данных.

Тело хранимой процедуры является составным оператором, т.е. совокупностью операторов, заключенных между служебными словами **BEGIN** и **END**. Наряду с SQL-операторами в составном операторе могут быть определены локальные переменные, курсоры, временные таблицы данных и исключительные ситуации. Они доступны только в пределах составного оператора и не видимы за его пределами. Время их существования ограничено периодом исполнения составного оператора. Локальные определения широко используются при разработке программных объектов.

При выполнении процедуры, формирующей результирующее множество, создается временная таблица - курсор (**CURSOR**). В курсор записывается результирующее множество. В дальнейшем пользователь может обрабатывать данные курсора по-своему усмотрению.

Одним из способов возврата результатов работы хранимых процедур является формирование результирующего множества. Данное множество формируется при выполнении оператора **SELECT**. Оно записывается во временную таблицу - курсор. Применение курсора в процедурах осуществляется путем последовательного выполнения следующих шагов:

- при помощи оператора **DECLARE** объявляется курсор для отдельного оператора **SELECT** или для отдельной процедуры;
- оператором **OPEN** производится открытие курсора;
- используя оператор **FETCH**, осуществляется установление указателя на требуемую запись курсора. При этом значения полей текущей записи присваиваются переменным, указываемым в операторе **FETCH**;
- в процессе перемещения указателя текущей записи курсора при выходе указателя за пределы курсора выдается предупреждение **row not found**;
- после того как курсор становится ненужным, он закрывается оператором **CLOSE**.

По умолчанию курсор закрывается автоматически в конце транзакции (операторы **COMMIT** или **ROLLBACK**). Если при объявлении курсора указана фраза **WITH HOLD** (с сохранением), то курсор закрывается только явным образом оператором **CLOSE**.

Хранимые функции являются разновидностью хранимых процедур. Они включены в состав программных объектов баз данных с целью наибольшего соответствия языкам программирования, например C или Pascal. Как и в этих языках программирования, каждая хранимая функция рассматривается в качестве выражения, формирующего одно единственное значение. Хранимые функции применяются для расширения функциональных возможностей операторов **SELECT** и ряда других SQL-операторов. Хранимые функции создаются оператором **CREATE FUNCTION**. Их модификация производится при помощи оператора **ALTER FUNCTION**.

Оператор **CREATE FUNCTION** несколько отличается от оператора **CREATE PROCEDURE**. Эти отличия состоят в следующем:

- в функциях допустимы только входные формальные параметры;
- после описания формальных параметров функции должен следовать оператор **RETURNS**. Данный оператор указывает тип возвращаемого функцией значения;
- в теле функции обязательно должен присутствовать оператор **RETURN**, который специфицирует возвращаемое функцией значение;
- функция не может формировать результирующее множество.

Вызов хранимой функции должен осуществляться там, где требуется выражение, формирующее значение. В связи с этим, функции могут непосредственно использоваться в выражениях. Например, они могут применяться в арифметических выражениях, вместо фактических параметров процедур или функций, а также прямо в составе SQL-операторов. Все эти качества позволяют в значительной степени расширить функциональные возможности языка SQL, как средства разработки приложений.

Триггеры - это один из видов программных объектов СУБД, поддерживаемых Microsoft SQL Server 2005. Каждый триггер связан с одной из таблиц данных, входящих состав базы данных. С каждой такой таблицей может быть связано несколько триггеров. Основное назначение триггеров состоит в

автоматическом использовании их в качестве реакции на некоторые события, происходящие с таблицами, с которыми связаны триггеры. Это свойство триггеров позволяет использовать их для реализации сложных форм ограничений целостности данных. Кроме того, рассматриваемое свойство превращает сервер из пассивного наблюдателя за происходящими изменениями данных в систему, оперативно реагирующую на такие изменения. Правила, в соответствии с которыми осуществляются активные действия сервера, определяются триггерами.

Триггеры, создаются оператором **CREATE TRIGGER**. Модификация триггеров производится при помощи оператора **ALTER TRIGGER**, а удаление – оператором **DROP TRIGGER**.

В СУБД Microsoft SQL Server 2005 триггеры могут быть определены для одного из приведенных ниже событий или сразу на несколько из них:

- добавление новой записи в таблицу. Данное событие возникает при выполнении оператора **INSERT**. Оно приводит к активизации триггера при добавлении новой записи в связанную таблицу;

- удаление записей. Это событие наступает в результате воздействия оператора **DELETE**. Данный оператор производит удаление записей из таблиц, с которыми связаны триггеры, что является причиной активизации триггера;

- модификация записей. Событие этого типа инициируется оператором **UPDATE**. Оно возникает при изменении значений любого из полей записей таблицы, с которой связан активизируемый триггер;

- изменение значений заданного списка полей таблицы. Это событие, как и предыдущее, возникает при выполнении оператора **UPDATE (UPDATE OF columns-list)**, но при модификации только заданных полей таблицы данных (**columns-list**).

Таким образом, триггеры автоматически запускаются при изменении содержимого таблицы данных, с которой они связаны.

Для каждого триггера должно быть определено время его выполнения - либо перед операторами **INSERT**, **DELETE**, **UPDATE** (предваряющий триггер), либо после них (завершающий триггер). Типичный пример использования предваряющих триггеров - проверка вводимых данных. Завершающие триггеры полезны в тех случаях, когда при модификации записей необходимо сравнивать исходные значения полей с их новыми значениями.

Операторы **INSERT**, **DELETE**, **UPDATE**, которые содержат служебное слово **WHERE** или подзапросы, как правило, воздействуют (добавляют, удаляют, модифицируют) на несколько записей таблицы. Когда при этом должен выполняться триггер? Каждый раз при изменении очередной записи или один раз после модификации всех записей? Для ответа на этот вопрос в Microsoft SQL Server 2005 реализованы триггеры двух уровней. Триггер первого уровня **Statement-level trigger** выполняется однократно после полного завершения одного из вышеуказанных операторов (**операторный триггер**) Триггер второго уровня **Row-level trigger** предназначен для многократного выполнения после каждого изменения одной из записей (строк) таблицы (**строчный триггер**).

Если при выполнении триггера возникает ошибка, тогда все действия осуществленные оператором, активизировавшим триггер, и действия, уже произведенные самим триггером, отменяются. В результате этого база данных возвращается в состояние, предшествующее началу выполнения вышеуказанного оператора.

Общий порядок применения триггеров состоит в последовательном выполнении следующих шагов при вызове на выполнение операторов **INSERT**, **DELETE** или **UPDATE**:

1. Активизируются предваряющие триггеры, если они есть.
2. Выполняются действия по проверке ограничений ссылочной целостности, установленной при определении схемы базы данных и возможно нарушенной действиями предваряющих триггеров.
3. Выполняется непосредственно операция, активизирующая триггер.
4. Активизируются завершающие триггеры, в случае если таковые имеются .

При работе триггеров часто возникает необходимость обращаться как к исходным значениям полей таблицы, так и к их обновленным значениям. Для этой цели используются фраза **REFERENCING** с параметрами **NEW AS** или **OLD AS**. С ее помощью вводятся псевдонимы для обращения к исходным и обновленным данным. В конечном же итоге смысл таких псевдонимов зависит от типа триггера и от событий, при возникновении которых активизируются триггеры. Назначение псевдонимов, вводимых фразой **REFERENCING**, приведено в табл. 4.1.

Таблица 4.1 – Назначение псевдонимов, вводимых фразой **REFERENCING** для обращения к исходным и обновленным записям при определении триггеров

Тип триггера		Параметр для REFERENCING	Операторы, активизирующие триггер		
			INSERT	DELETE	UPDATE
Строчный	Предваряющий	NEW AS	Для обращения к полям вводимой записи	--	Для обращения к новым значениям модифицируемых полей записи
		OLD AS	--	Для обращения к полям удаляемой записи	Для обращения к исходным значениям модифицируемых полей записи
	Завершающий	NEW AS	Для обращения к полям введенной записи	--	Для обращения к новым значениям модифицируемых полей записи

		OLD AS	--	Для обращения к полям удаленной записи	Для обращения к исходным значениям модифицируемых полей записи
Операторный	Завершающий	NEW AS	Для обращения к временной таблице, содержащей введенные записи	--	Для обращения к временной таблице, содержащей модифицированные записи
		OLD AS	--	Для обращения к временной таблице, содержащей удаленные записи	Для обращения к временной таблице, содержащей исходные значения обновленных записей

ВЫПОЛНЕНИЕ РАБОТЫ

I. Создание и использование хранимых процедур

Для получения доступа к перечню хранимых процедур нужно в списке объектов базы данных открыть пункт Programmability и в нем открыть пункт Stored Procedures. В результате появится список пользовательских хранимых процедур (если такие процедуры были созданы ранее). Пункт System Stored Procedures открывать не нужно.

1. Создание хранимой процедуры, реализующей выборку данных из таблиц Договоры, ЮридическиеЛица, ФизическиеЛица.

Такая хранимая процедура может обеспечить вывод связанных данных, находящихся в нескольких таблицах. Для создания хранимой процедуры следует выполнить следующую последовательность действий

1. Щелкнуть правой кнопкой мыши по пункту Stored Procedures и в появившемся меню выбрать пункт New Stored Procedure.... В результате будет создан запрос, содержащий «заготовку» хранимой процедуры. Пример такой «заготовки» приведен на рисунке 4.1

```
-- =====
-- Template generated from Template Explorer using:
-- Create Procedure (New Menu).SQL
--
-- Use the Specify Values for Template Parameters
-- command (Ctrl-Shift-M) to fill in the parameter
-- values below.
--
-- This block of comments will not be included in
-- the definition of the procedure.
-- =====
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
-- =====
-- Author:      <Author,,Name>
-- Create date: <Create Date,,>
-- Description: <Description,,>
-- =====
CREATE PROCEDURE <Procedure_Name, sysname, ProcedureName>
-- Add the parameters for the stored procedure here
<@Param1, sysname, @p1> <Datatype_For_Param1, , int> = <Default_Value_For_Param1, , 0>,
<@Param2, sysname, @p2> <Datatype_For_Param2, , int> = <Default_Value_For_Param2, , 0>
AS
BEGIN
-- SET NOCOUNT ON added to prevent extra result sets from
-- interfering with SELECT statements.
SET NOCOUNT ON;

-- Insert statements for procedure here
SELECT <@Param1, sysname, @p1>, <@Param2, sysname, @p2>
END
GO
```

Рисунок 4.1

2. Эту «заготовку» нужно изменить, сформировав следующий текст процедуры (рисунок 4.2)

```
set ANSI_NULLS ON
set QUOTED_IDENTIFIER ON
go

CREATE PROCEDURE [dbo].[sp_Договоры]
AS
BEGIN
    SET NOCOUNT ON
    SELECT *
    FROM (Договоры LEFT JOIN ЮридическиеЛица ON
        Договоры.КодПоставщика=ЮридическиеЛица.КодПоставщика)
    LEFT JOIN ФизическиеЛица ON
        Договоры.КодПоставщика=ФизическиеЛица.КодПоставщика
END
```

Рисунок 4.2

3. Для создания этой хранимой процедуры нужно нажать кнопку Execute на панели инструментов. В том случае, если запрос выполнен успешно, в окне Messages появится сообщение Command(s) completed successfully. В этом случае окно запроса с текстом хранимой процедуры можно закрыть, причем запрос сохранять в виде файла не нужно. Хранимая процедура должна появиться в списке хранимых процедур. Если процедура отсутствует в списке, нужно щелкнуть правой кнопкой мыши по пункту Stored Procedures и в появившемся меню выбрать пункт Refresh
4. Для проверки работы хранимой процедуры нужно создать новый запрос и ввести оператор вызова хранимой процедуры (рисунок 4.3). В результате выполнения хранимой процедуры на экран будет выведена таблица, содержащая результат запроса, реализованного в хранимой процедуре. Этот запрос можно закрыть, не сохраняя. Другим способом выполнения хранимой процедуры является ее выбор в списке хранимых процедур правой кнопкой мыши и выбор в появившемся меню пункта Execute Stored Procedure....

```
use delivery
exec sp_Договоры
```

Рисунок 4.3

5. При необходимости внесения изменений в текст процедуры, ее можно открыть в режиме редактирования, для чего нужно щелкнуть правой кнопкой мыши по имени процедуры в списке процедур и в появившемся меню выбрать пункт Modify. В результате на экран будет выведен запрос, содержащий текст хранимой процедуры. Если в текст процедуры вносились изменения и эти изменения нужно сохранить, то запрос, содержащий текст хранимой процедуры, нужно выполнить. После успешного выполнения запрос в виде файла сохранять не нужно. Для проверки изменения хранимой процедуры ее нужно вновь открыть в режиме редактирования.

2. Создание хранимой процедуры, обеспечивающей формирование агрегированных данных по поставкам для указанного интервала календарных дат

Последовательность действий при создании процедуры аналогична описанной выше. Особенностью создаваемой процедуры является наличие параметров. Текст процедуры приведен на рисунке 4.4

```
set ANSI_NULLS ON
set QUOTED_IDENTIFIER ON
go

CREATE PROCEDURE [dbo].[sp_dgvр_agr]
    @var1 datetime,
    @var2 datetime
AS
BEGIN
    SET NOCOUNT ON;
    SELECT Договоры.НомерДоговора, ДатаДоговора, SUM(Количество) , SUM(Количество*Цена)
    FROM Договоры LEFT JOIN Поставлено ON
        Договоры.НомерДоговора=[Поставлено].[НомерДоговора]
    WHERE ДатаДоговора BETWEEN @var1 AND @var2
    GROUP BY Договоры.НомерДоговора, ДатаДоговора
END
```

Рисунок 4.4

Процедуру нужно сохранить и затем запустить для проверки работоспособности. Используемый для этого запрос может иметь вид, приведенный на рисунке 4.5 или 4.6

```
use delivery
exec sp_dgvр_agr '1999/01/01', '1999/10/31'
```

Рисунок 4.5

```
use delivery
exec sp_dgvр_agr '19990101', '19991031'
```

Рисунок 4.6

3. Создание хранимой процедуры, которая реализует различные операции модификации данных для таблицы Договоры.

Такая хранимая процедура должна обеспечить возможность создания нового договора или изменения параметров уже существующего договора или удаления уже существующего договора. Текст такой процедуры может иметь вид, приведенный на рисунке 4.7. Процедуру нужно сохранить и затем запустить для проверки работоспособности. На рисунках 4.8 – 4.10 приведены варианты запуска процедуры в режиме создания договора, его модификации и удаления соответственно. При запуске процедуры в режиме модификации или

удаления договора нужно четко определить номер договора, для которого выполняются эти операции. После каждого выполнения процедуры необходимо проверять состояние базы данных

```
set ANSI_NULLS ON
set QUOTED_IDENTIFIER ON
go

CREATE PROCEDURE [dbo].[sp_dgvr_mdf]
    @Action char(1), @nom_dgvr int,
    @dgvr_date datetime, @dgvr_kod_post int, @dgvr_comment text
AS
SET NOCOUNT ON
BEGIN
    IF @Action='I'
    BEGIN
        print 'insert'
        INSERT INTO Договоры (ДатаДоговора, КодПоставщика, Комментарий)
            VALUES (GETDATE(), @dgvr_kod_post, @dgvr_comment)
    END
    ELSE
    IF @Action='U'
    BEGIN
        print 'update'
        UPDATE Договоры SET ДатаДоговора=@dgvr_date,
            КодПоставщика=@dgvr_kod_post,
            Комментарий=@dgvr_comment
            WHERE НомерДоговора=@nom_dgvr
    END
    ELSE
    IF @Action='D'
    BEGIN
        print 'delete'
        DELETE FROM Договоры WHERE НомерДоговора=@nom_dgvr
    END
END
```

Рисунок 4.7

```
use delivery
exec sp_dgvr_mdf 'I', 0, '2008/12/16', 2, ''
```

Рисунок 4.8

```
use delivery
exec sp_dgvr_mdf 'U', 8, '2008/12/31', 2, '88888888'
```

Рисунок 4.9

```
use delivery
exec sp_dgvr_mdf 'D', 8, '2008/12/31', 0, ''
```

Рисунок 4.10

II. Создание и использование триггеров

1. Создание триггера, контролирующего наличие даты договора на поставку продукции

Предположим, что при вводе данных в таблицу Договоры, в которой хранится информация о договорах на поставку продукции, поле ДатаДоговора, в котором хранится дата заключения договора, должно быть обязательно заполнено, причем в том случае, если при вводе нового договора это поле остается незаполненным, в него должна быть автоматически записана текущая дата. Эту задачу можно решить разными средствами, в том числе и с помощью триггера.

Для получения доступа к перечню триггеров уровня таблицы нужно в списке таблиц открыть список объектов требуемой таблицы (в данном случае – Договоры) и в этом списке открыть пункт Triggers. В результате появится список триггеров (если триггеры для таблицы уже были созданы ранее).

Для создания нового триггера выполним следующие действия.

1. Щелкнуть правой кнопкой мыши по пункту Triggers и в появившемся меню выбрать пункт New Trigger.... В результате будет создан запрос, содержащий «заготовку» триггера. Эту «заготовку» нужно изменить, введя текст триггера, приведенный на рисунке 4.11

```
set ANSI_NULLS ON
set QUOTED_IDENTIFIER ON
go

CREATE TRIGGER [not_null_date] ON [dbo].[Договоры]
AFTER INSERT NOT FOR REPLICATION AS

BEGIN
    SET NOCOUNT ON;

    DECLARE @new_dgvr_date datetime
    DECLARE @new_dgvr_nomer int
    SELECT @new_dgvr_nomer=НомерДоговора,@new_dgvr_date=ДатаДоговора FROM inserted
    IF @new_dgvr_date IS NULL
    BEGIN
        UPDATE Договоры SET ДатаДоговора=GETDATE() WHERE НомерДоговора=@new_dgvr_nomer
    END
END
```

Рисунок 4.11

2. Для создания этого триггера нужно нажать кнопку Execute на панели инструментов. В том случае, если запрос выполнен успешно, в окне Messages появится сообщение Command(s) completed successfully. В этом случае окно запроса с текстом триггера можно закрыть, причем запрос сохранять в виде файла не нужно. Триггер должен появиться в списке триггеров таблицы. Если триггер отсутствует в списке, нужно щелкнуть правой кнопкой мыши по пункту Triggers и в появившемся меню выбрать пункт Refresh

3. Для проверки работы триггера нужно добавить новый договор в список договоров. Это можно сделать, например, с помощью запроса, приведенного на рисунке 4.12. После успешного выполнения запроса нужно проверить состояние таблицы Договоры. В поле ДатаДоговора записи, соответствующей новому договору должна быть записана текущая календарная дата.

```
use delivery
INSERT INTO Договоры (КодПоставщика, Комментарий)
VALUES (1, '')
```

Рисунок 4.12

2. Создание триггера, контролирующего наличие данных о поставщике - юридическом лице

В базе данных хранится как общая информация о поставщиках, так и информация, которая относится только к поставщикам – физическим лицам или поставщикам – юридическим лицам. Каждый поставщик может быть или юридическим, или физическим лицом. Это значит, что одновременное наличие данных о поставщике в таблицах ЮридическиеЛица и ФизическиеЛица не допускается с точки зрения требований логики управления бизнесом. Таким образом, возникает необходимость сложного контроля отношений ссылочной целостности. Для решения этой задачи создадим триггер, который при вводе информации в таблицу ФизическиеЛица будет контролировать наличие кода соответствующего поставщика в таблице ЮридическиеЛица и блокировать ввод данных о поставщике как о физическом лице в том случае, если уже имеются данные об этом поставщике как о юридическом лице.

Последовательность действий при создании триггера аналогична описанной ранее. Текст триггера приведен на рисунке 4.13. После ввода текста, триггер нужно сохранить, выполнив запрос. Затем нужно проверить работоспособность триггера. Для этого попробуем добавить в таблицу ФизическиеЛица данные о поставщике, который уже является юридическим лицом. Это можно сделать, например, с помощью запроса, приведенного на рисунке 4.14. Такой запрос должен инициировать выполнение триггера. Это, в частности, выражается в результате выполнения запроса, который приведен на рисунке 4.15. Кроме того, контроль состояния данных в таблице ФизическиеЛица должен подтвердить то, что новые данные в этой таблице не появились.

```

set ANSI_NULLS ON
set QUOTED_IDENTIFIER ON
go

CREATE TRIGGER [check_yur_1] ON [dbo].[ФизическиеЛица]
    AFTER INSERT NOT FOR REPLICATION AS

BEGIN
    SET NOCOUNT ON;
    declare @new_fiz_code int
    declare @var1 int
    select @new_fiz_code=КодПоставщика from inserted
    print 'Попытка добавить новое физическое лицо с кодом' + str(@new_fiz_code)
    select @var1=count(КодПоставщика) from ЮридическиеЛица where КодПоставщика=@new_fiz_code
    IF @var1>0
        BEGIN
            DECLARE @DBID INT;
            SET @DBID = DB_ID();
            DECLARE @DBNAME NVARCHAR(128);
            SET @DBNAME = DB_NAME();
            RAISERROR
            (N'The current database ID is:%d, the database name is: %s.',
            10,
            1,
            @DBID,
            @DBNAME);
            print 'Поставщик с кодом ' + str(@new_fiz_code) + ' уже является юридическим лицом'
            ROLLBACK
        END
    ELSE
        PRINT 'Новое физическое лицо с кодом ' + str(@new_fiz_code) + ' успешно добавлено'
END

```

Рисунок 4.13

```

use delivery
INSERT INTO ФизическиеЛица
    VALUES (2, 'Макаров', 'Олег', 'Петрович', '00123987');

```

Рисунок 4.14

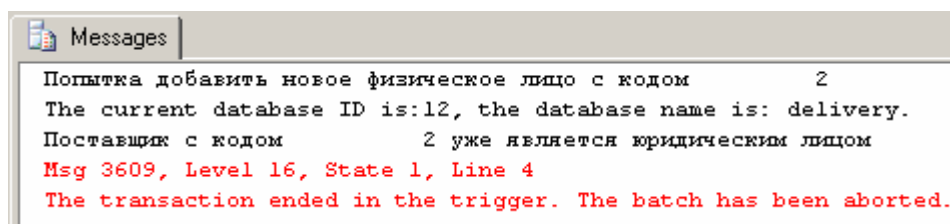


Рисунок 4.15

Сохранение результатов работы

Сохранить файл базы данных

Требования к отчету:

- 1) кратко описать основные этапы выполнения задания
- 2) описать созданные хранимые процедуры и триггеры и результаты их использования

Лабораторная работа 5

Создание и использование представлений (view)

Представления (view) – это одно из мощных средств языка SQL, предназначенное для реализации механизма подсхем пользователей базы данных. Представления позволяют скрыть от пользователей схему базы данных. Они представляют собой хранимые в базе данных запросы, выраженные операторами **SELECT**. На базе одних представлений могут быть созданы новые представления, которые наследуют все свойства базовых представлений. Формировать представления могут пользователи с привилегиями **SELECT** для используемых в представлениях таблиц (базовых таблиц).

Для пользователя представления предстают как объекты очень похожие на таблицы данных. Это выражается тем, что:

- обращение к представлениям осуществляется также как и к таблицам;
- ко всем представлениям применим оператор **SELECT**;
- для некоторых представлений могут применяться операторы **INSERT**, **UPDATE** и **DELETE**.

Однако в соответствие со стандартом ANSI SQL/89 в Microsoft SQL Server 2005 таблицы данных и представления имеют некоторые различия:

- запрос, именованный через представление выполняется только в момент обращения к представлению;
- для представления невозможно определить ограничения целостности и первичный ключ;
- в операторе **SELECT**, на базе которого создается представление, нельзя устанавливать сортировку его результатов;
- не ко всем представлениям могут применяться операторы **INSERT**, **UPDATE** и **DELETE**.

Представление может быть модифицировано (т.е. по отношению к нему можно использовать операторы **INSERT**, **UPDATE** и **DELETE**) в том, и только в том случае, если для оператора **SELECT**, на базе которого создано представление, выполняются каждое из следующих специфических условий:

- не используется служебное слово **DISTINCT**;
- при выполнении запроса данные извлекаются только из таблицы;
- в списке полей этого оператора отсутствуют арифметические выражения. Элементами списка могут быть только поля базовой таблицы или базового представления. В свою очередь на поля этого представления накладывается такое же ограничение;
- в запросе не применяются подзапросы;
- для результирующих данных не определено группирование.

Параметр **WITH CHECK OPTION** используется при создании обновляемых представлений, ключевые слова **LOCAL** и **CASCADED** определяют границы проверок в случае, если представление определено с

помощью другого представления. LOCAL ограничивает проверки текущим представлением, в то время как CASCADED вызывает проверки для родительских представлений. CASCADED используется по умолчанию.

ВЫПОЛНЕНИЕ РАБОТЫ

1. Создание представления, позволяющего при просмотре списка договоров видеть название поставщика.

Для создания представления следует выполнить следующую последовательность действий

1. Щелкнуть правой кнопкой мыши по пункту Views и в появившемся меню выбрать пункт New View...
2. В появившемся списке таблиц выбрать таблицы Договоры, Поставщики, ЮридическиеЛица, ФизическиеЛица. Список таблиц закрыть. В результате появится графическое изображение таблиц, используемых в качестве источника данных для представления, и связей между ними (рисунок 5.1)

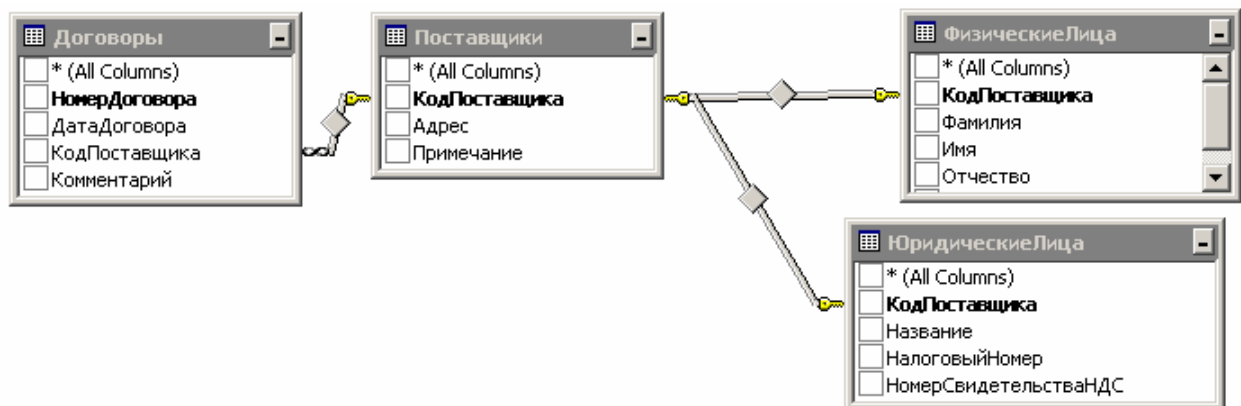


Рисунок 5.1

3. Щелкнуть правой кнопкой мыши по связи между таблицами Поставщики и ФизическиеЛица и выбрать пункт Select All Rows from Поставщики. Щелкнуть правой кнопкой мыши по связи между таблицами Поставщики и ЮридическиеЛица и выбрать пункт Select All Rows from Поставщики. В результате связи примут вид, приведенный на рисунке 5.2

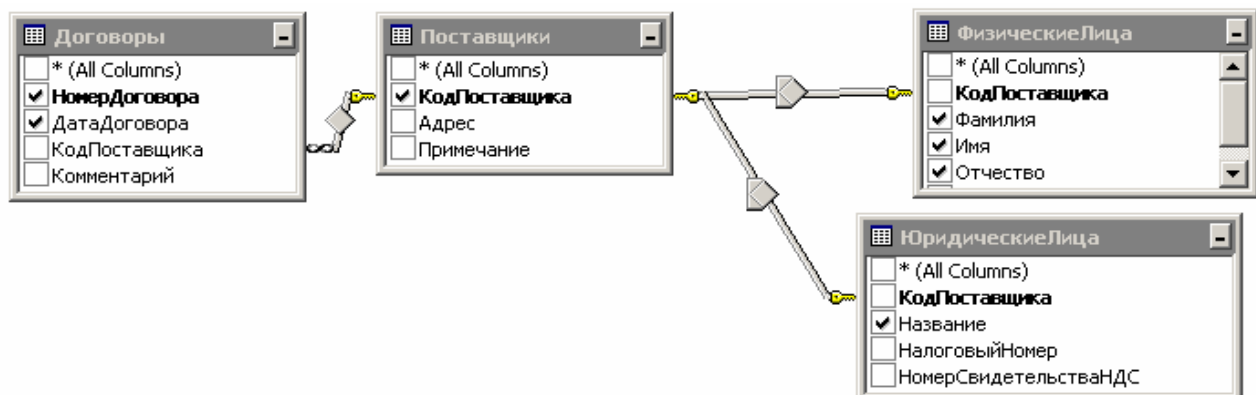


Рисунок 5.2

4. Выбрать поля таблиц, включаемые в результат запроса, поставив отметки для соответствующих полей (рисунок 5.2). В результате текст запроса представления будет иметь вид, приведенный на рисунке 5.3. Нажав на панели инструментов кнопку Execute SQL, можно получить результат запроса. Этот результат имеет определенный недостаток – данные поставщиков – юридических лиц и физических лиц находятся в разных полях. Этот недостаток можно исправить, изменив текст запроса (рисунок 5.4).

```
SELECT  dbo.Договоры.НомерДоговора, dbo.Договоры.ДатаДоговора, dbo.Поставщики.КодПоставщика, dbo.ЮридическиеЛица.Название,
        dbo.ФизическиеЛица.Фамилия, dbo.ФизическиеЛица.Имя, dbo.ФизическиеЛица.Отчество
FROM    dbo.Договоры INNER JOIN
        dbo.Поставщики ON dbo.Договоры.КодПоставщика = dbo.Поставщики.КодПоставщика LEFT OUTER JOIN
        dbo.ЮридическиеЛица ON dbo.Поставщики.КодПоставщика = dbo.ЮридическиеЛица.КодПоставщика LEFT OUTER JOIN
        dbo.ФизическиеЛица ON dbo.Поставщики.КодПоставщика = dbo.ФизическиеЛица.КодПоставщика
```

Рисунок 5.3

```
SELECT  dbo.Договоры.НомерДоговора, dbo.Договоры.ДатаДоговора, dbo.Поставщики.КодПоставщика,
        ISNULL(dbo.ЮридическиеЛица.Название + SPACE(30),
        dbo.ФизическиеЛица.Фамилия + dbo.ФизическиеЛица.Имя + dbo.ФизическиеЛица.Отчество) AS поставщик
FROM    dbo.Договоры INNER JOIN
        dbo.Поставщики ON dbo.Договоры.КодПоставщика = dbo.Поставщики.КодПоставщика LEFT OUTER JOIN
        dbo.ЮридическиеЛица ON dbo.Поставщики.КодПоставщика = dbo.ЮридическиеЛица.КодПоставщика LEFT OUTER JOIN
        dbo.ФизическиеЛица ON dbo.Поставщики.КодПоставщика = dbo.ФизическиеЛица.КодПоставщика
```

Рисунок 5.4

5. Сохранить представление с именем View_1
6. Проверить работу представления, для чего щелкнуть правой кнопкой мыши по имени представления и в появившемся меню выбрать команду «Open View». Проанализировать информацию, которая выводится с помощью представления

2. Создание обновляемого представления, позволяющего пользователю работать с ограниченными данными о поставщиках.

Предположим, что для определенных пользователей должна быть доступна не вся общая информация о поставщиках (хранящаяся в таблице Поставщики), а только информация о коде и адресе поставщика. При этом пользователь должен иметь возможность видеть данные поставщика как субъекта предпринимательской деятельности (для юридических лиц – название, для физических – фамилия, имя, отчество). При вводе нового поставщика вводится только информация о коде и адресе, а при необходимости корректировки данных пользователь может изменить только адрес поставщика.

Последовательность действий при создании этого представления аналогична описанной выше. Графическое изображение созданного представления приведено на рисунке 5.5. Текст запроса представления приведен на рисунке 5.6. Созданное представление сохранить с именем View_2. После сохранения проверить работу представления.

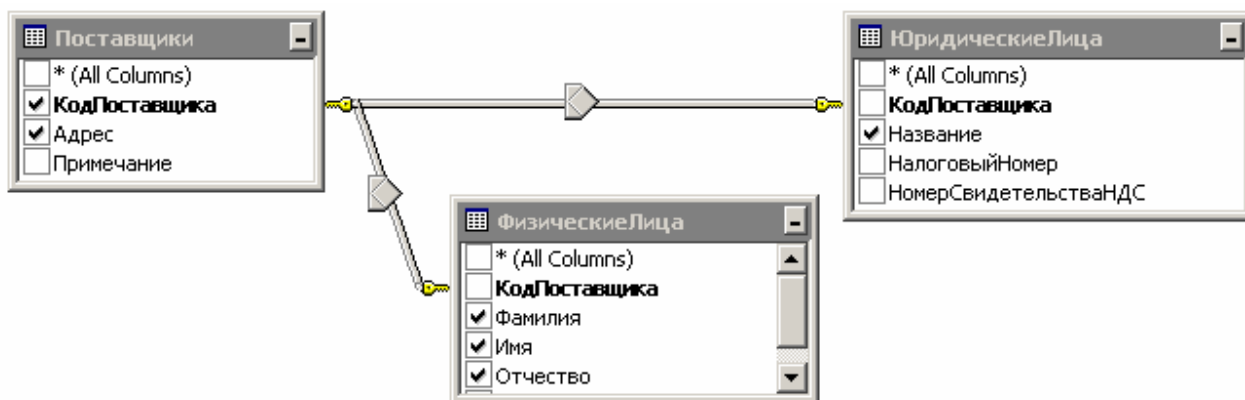


Рисунок 5.5

```
SELECT  dbo.Поставщики.КодПоставщика, dbo.Поставщики.Адрес, dbo.ФизическиеЛица.Фамилия, dbo.ФизическиеЛица.Имя,
        dbo.ФизическиеЛица.Отчество, dbo.ЮридическиеЛица.Название
FROM    dbo.Поставщики LEFT OUTER JOIN
        dbo.ФизическиеЛица ON dbo.Поставщики.КодПоставщика = dbo.ФизическиеЛица.КодПоставщика LEFT OUTER JOIN
        dbo.ЮридическиеЛица ON dbo.Поставщики.КодПоставщика = dbo.ЮридическиеЛица.КодПоставщика
```

Рисунок 5.6

Для добавления данных в таблицу Поставщики с помощью представления нужно выполнить следующую последовательность действий.

1. Создать новый запрос, нажав кнопку New Query на панели инструментов
2. Ввести текст запроса (пример приведен на рисунке 5.7). Код поставщика нужно указать с учетом состояния данных в таблице Поставщики

```
use delivery
insert into view_2 (КодПоставщика, Адрес ) values (6, 'г.Ахтырка, ул.Ленина, 25')
```

Рисунок 5.7

3. Выполнить запрос
4. В случае успешного выполнения запроса проверить наличие записи о новом поставщике в таблице Поставщики

Сохранение результатов работы

Сохранить файлы базы данных

Требования к отчету:

- 1) кратко описать основные этапы выполнения задания
- 2) описать созданные представления и результаты их использования

Лабораторная работа 6

Изучение основ работы со средствами контроля ссылочной целостности данных (Referential Integrity)

Перед началом выполнения работы рассмотрим краткий обзор основных положений, связанных с понятием ссылочной целостности данных (Referential Integrity) и особенностями использования механизмов ссылочной целостности.

Связывание строк таблиц реляционной базы данных выполняется с помощью *первичного* (primary) и *внешнего* (foreign) ключей. Разработчик базы данных должен определить правила связывания данных в разных таблицах, выделив в них одно или более полей в качестве первичного или внешнего ключа. Как известно, первичный ключ позволяет однозначно идентифицировать любую строку таблицы. При выборе столбцов, которые будут входить в состав первичного ключа, необходимо следовать требованиям *уникальности* и *минимальности*.

Следующий шаг в связывании таблиц — определение *внешнего ключа*. Внешний ключ создается в таблице, поля которой ссылаются на строки главной таблицы. Для каждой строки зависимой таблицы необходимо, чтобы значению внешнего ключа было сопоставлено значение первичного ключа. То есть нельзя вставлять в зависимую таблицу строки со значением внешнего ключа, не определенного в главной таблице. Однако допускается, что значение внешнего ключа в зависимой таблице будет не определено, т. е. внешний ключ будет хранить значение Null. Впоследствии это значение может быть изменено на корректное значение, соответствующее значению внешнего ключа в главной таблице.

В отличие от первичного ключа, внешний ключ не должен быть уникальным. То есть в зависимой таблице может существовать множество строк, имеющих одинаковые значения для полей, сконфигурированных в качестве внешнего ключа. При этом разрешается устанавливать дополнительные ограничения целостности на поля, включенные во внешний ключ. Пользователь может установить ограничение целостности UNIQUE и тем самым гарантировать уникальность значений внешнего ключа.

После того, как первичный и внешний ключи будут связаны, на данные в зависимой таблице будут наложены ограничения на значения полей, определенных в качестве внешнего ключа. При этом возникает необходимость как-то согласовывать изменения ключевых полей, осуществляемые в главной таблице, со значениями в зависимой таблице. Если не выполнять никаких дополнительных действий, то возможно нарушение целостности данных (в частности, нарушение целостности данных может выражаться в том, что строки в зависимых таблицах окажутся *«потерянными»*, т.е. для них не будет сопоставлена ни одна строка главной таблицы).

Во избежание подобных проблем в SQL Server реализованы специальные механизмы, обеспечивающие автоматическую поддержку целостности данных. При попытке изменения (командой UPDATE) значения первичного ключа в главной таблице ядро SQL Server может вести себя следующим образом:

- **принудительное установление (Relation, Set Null).** Когда значение первичного ключа главной таблицы изменяется, то SQL Server автоматически устанавливает значения внешних ключей во всех связанных строках в неопределенное значение (NULL). При этом теряется информация о том, с какой строкой главной таблицы были связаны строки зависимой таблицы. При изменении в главной таблице более одной строки в зависимых таблицах может образоваться несколько наборов строк с неопределенным значением внешнего ключа. Определить, какая строка зависимой таблицы с какой строкой главной таблицы была связана, станет невозможно. Разновидностью данного механизма можно считать механизм **Set Default**. В этом случае SQL Server автоматически устанавливает значения внешних ключей во всех связанных строках в некоторое заранее заданное значение;

- **ограничение (Restrict, No Action).** В этом режиме SQL Server будет отвергать попытки изменения значения первичного ключа, если в зависимой таблице имеется хоть одна строка, связанная с изменяемой строкой главной таблицы. Изменение разрешается только в том случае, если ни в одной зависимой таблице не имеется ни одной строки, значение внешнего ключа которой совпадает со значением изменяемого первичного ключа. В общем случае, чтобы изменить значение первичного ключа, пользователь должен сам позаботиться о предварительном изменении значений связанных внешних ключей, например, установив их в неопределенное значение (NULL);

- **каскадирование (Cascading).** Это самый удобный и гибкий режим, обеспечивающий автоматическое соблюдение целостности данных. При изменении значения первичного ключа в главной таблице SQL Server 2000 будет автоматически изменять значения всех связанных внешних ключей во всех строках зависимых таблиц. Эти изменения окажутся незаметны для пользователей. Все имеющиеся связи будут автоматически сохранены.

Мы рассмотрели различные варианты поведения системы при попытке изменения значений первичного ключа. Однако необходимо также рассмотреть возможность удаления строк главной таблицы. Если в зависимых таблицах с первичным ключом удаляемой строки не была связана ни одна строка, то проблем нет. Если же такие строки в зависимых таблицах существуют, то необходимо выполнить удаление таким образом, чтобы обеспечить целостность данных. Возможны следующие варианты поведения SQL Server при удалении строк из главной таблицы:

- **принудительное установление (Relation, Set Null).** При удалении первичного ключа SQL Server будет автоматически устанавливать для всех связанных внешних ключей неопределенное значение (NULL). Впоследствии такие строки могут быть удалены вручную или связаны с другим первичным ключом. Разновидностью данного механизма можно считать механизм **Set Default**. В этом случае SQL Server автоматически устанавливает значения внешних ключей во всех связанных строках в некоторое заранее заданное значение;

- **ограничение (Restrict, No Action).** Перед тем, как станет возможным удаление строки в главной таблице, ни в одной зависимой таблице не должно быть строки, имеющей то же значение внешнего ключа, что и первичный ключ удаляемой строки. Пользователь обязан либо удалить такие строки из зависимой таблицы, либо установить для них значение внешнего ключа в неопределенное значение (NULL), либо связать его с любым другим первичным ключом главной таблицы.

- **каскадирование (Cascading).** В этом режиме система станет автоматически удалять все связанные строки из зависимых таблиц. Этот метод требует от пользователя минимального количества усилий.

ВЫПОЛНЕНИЕ РАБОТЫ

Внимание! При выполнении работы данные, хранящиеся в базе данных, будут изменяться. В связи с этим рекомендуется использовать не основную базу данных (delivery), а некоторую временную. В качестве такой базы данных можно использовать базу данных, созданную при выполнении лабораторной работы 2 или создать новую базу данных, используя запросы, разработанные при выполнении лабораторной работы 2. Таким образом, перед началом выполнения работы базу данных нужно создать или подключить.

I. Изучение особенностей работы механизма ссылочной целостности No Action

Рассмотрим особенности работы механизма ссылочной целостности No Action на примере отношений между таблицами Поставщики и Договоры, Поставщики и ФизическиеЛица, Поставщики и ЮридическиеЛица. Эти таблицы связаны между собой по полю КодПоставщика. В этой связи таблица Поставщики является родительской, а таблицы Договоры, ЮридическиеЛица, ФизическиеЛица – дочерними. Для изучения особенностей работы механизма ссылочной целостности выполним следующую последовательность действий.

1. В списке таблиц выбрать таблицу Поставщики, щелкнув по ней правой кнопкой мыши. В появившемся меню выбрать пункт Modify. В результате будет получен доступ к редактированию структуры таблиц.
2. Щелкнуть правой кнопкой мыши по любому полю таблицы и в появившемся меню выбрать пункт Relationships... В результате на экране появится окно Foreign Key Relationships (рисунок 6.1)

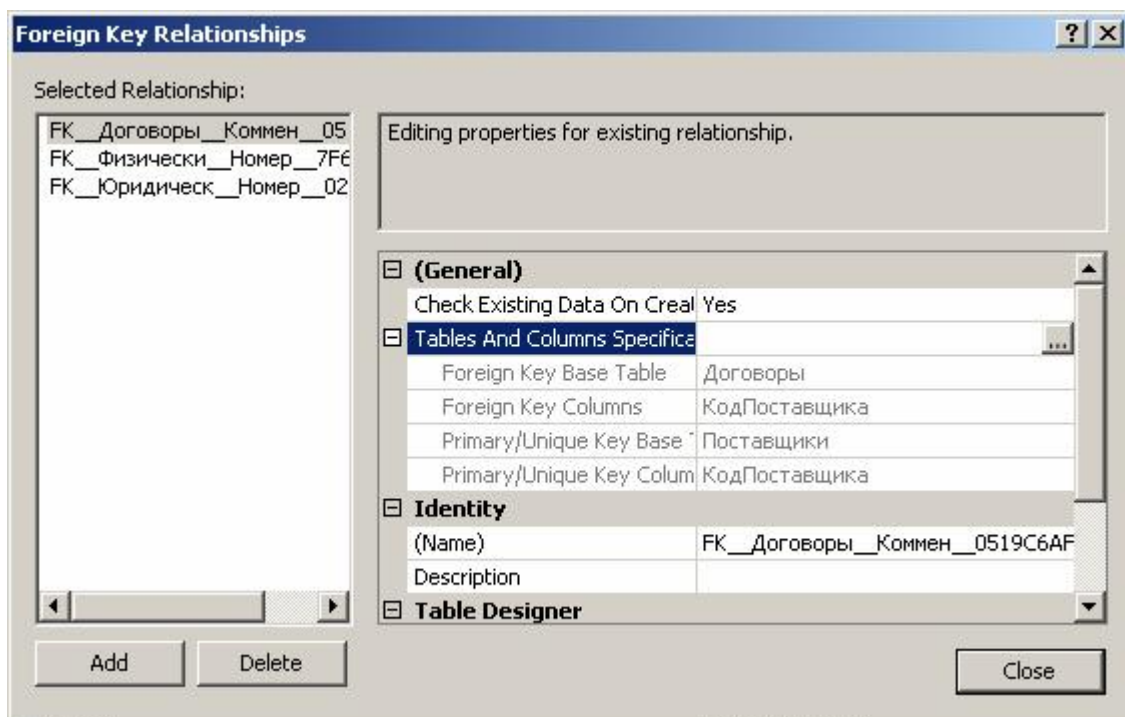


Рисунок 6.1

3. Выбрать связь, соответствующую связи между таблицами Поставщики и Договоры (рисунок 6.1). Раскрыв пункт Tables And Columns Specification можно увидеть информацию, показывающую, какие таблицы связаны и какие ключи при этом были использованы (рисунок 6.1).
4. Далее перейдем к рассмотрению механизмов ссылочной целостности, используемых при удалении записей в таблице Поставщики или изменения ключевого значения (т.е. значения поля КодПоставщика). Раскроем пункт INSERT And UPDATE Specification (рисунок 6.1). Как видно, механизм ссылочной целостности No Action установлен по умолчанию. Точно также нужно проверить механизм ссылочной целостности для связей между таблицами Поставщики и ФизическиеЛица и Поставщики и ЮридическиеЛица. Окно Foreign Key Relationships нужно закрыть. Также нужно закрыть окно, обеспечивающее доступ к структуре таблицы Поставщики.

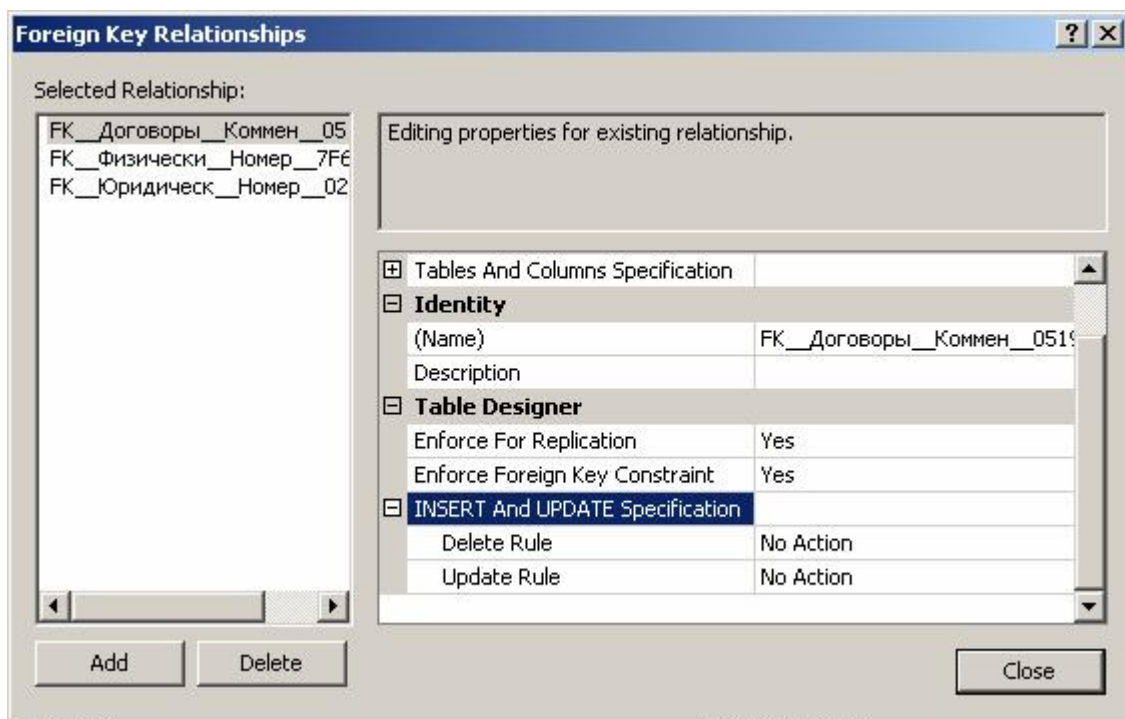


Рисунок 6.2

5. Открыть таблицу Поставщики в режиме просмотра/редактирования данных. Для этого таблицу нужно выбрать в списке таблиц, щелкнув по ней правой кнопкой мыши и в появившемся меню выбрать пункт Open Table. Аналогично нужно открыть таблицы Договоры, ЮридическиеЛица и ФизическиеЛица.
6. Предположим, что в силу каких-то причин необходимо удалить поставщика с кодом 4. Выбрав соответствующую запись в таблице Поставщики, нажмите правую кнопку мыши и в меню выберите пункт Delete. Затем подтвердите удаление записи. После этого на экран будет выведено окно (рисунок 6.3), информирующее пользователя о том, что удаление записи невозможно, т.к. на эту запись ссылаются записи в связанных таблицах.

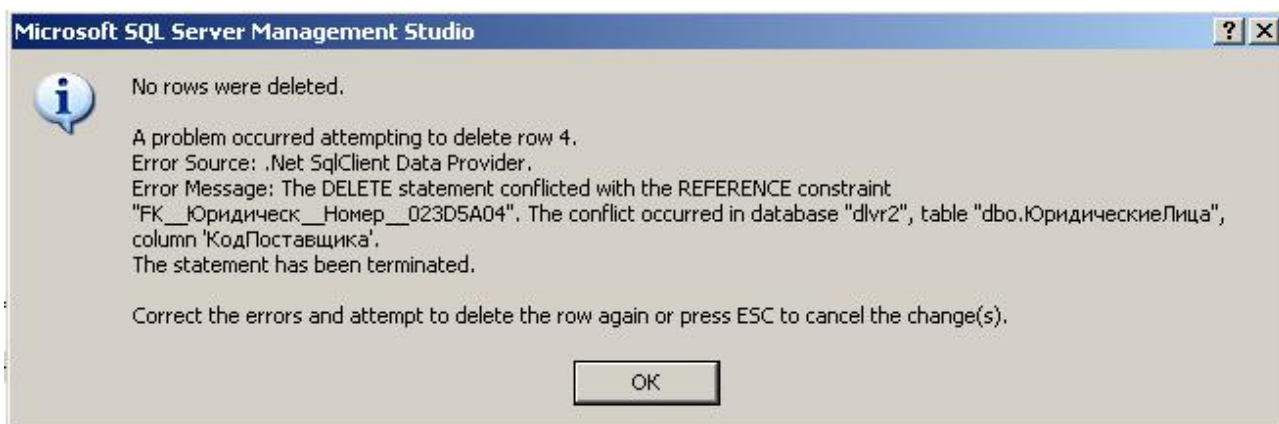


Рисунок 6.3

7. Таким образом, для того, чтобы удалить данного поставщика, нужно предварительно удалить все связанные с ним данные. Для этого нужно удалить соответствующую запись из таблицы ЮридическиеЛица и проверить наличие договоров с этим поставщиком в таблице Договоры. Если такие договоры есть, их тоже нужно удалить (при этом нужно иметь в виду, что может возникнуть необходимость удаления и содержимого этих договоров). После этого нужно попытаться повторить попытку удаления поставщика с кодом 4. Если связанных с ним данных нет, поставщик будет удален.
8. Предположим, что в силу каких-то причин возникла необходимость для поставщика с кодом 5 изменить код на 7. Выбрав соответствующую запись в таблице Поставщики, измените код поставщика с 5 на 7. Затем попытайтесь перейти на предыдущую запись. После этого на экран будет выведено окно (рисунок 6.4), информирующее пользователя о том, что изменение данных невозможно, т.к. на этот код ссылаются записи в связанных таблицах. Поскольку договоры с этим поставщиком отсутствуют, ссылка на него есть только в таблице ФизическиеЛица. Удалив эту запись, затем повторите попытку изменения кода поставщика с 5 на 7. Теперь эта операция должна пройти успешно. После этого нужно проверить содержимое таблиц и таблицы закрыть



Рисунок 6.4

II. Изучение особенностей работы механизма ссылочной целостности Cascade

Рассмотрим особенности работы механизма ссылочной целостности Cascade на примере отношений между таблицами Поставщики и Договоры, Поставщики и ФизическиеЛица, Поставщики и ЮридическиеЛица, Договоры и Поставлено. Доступ к связям выполняется также, как описано выше. Для изучения особенностей работы механизма ссылочной целостности выполним следующую последовательность действий.

1. Изменим механизмы ссылочной целостности для связей между всеми таблицами на Cascade. Пример результата изменения приведен на рисунке 6.5.

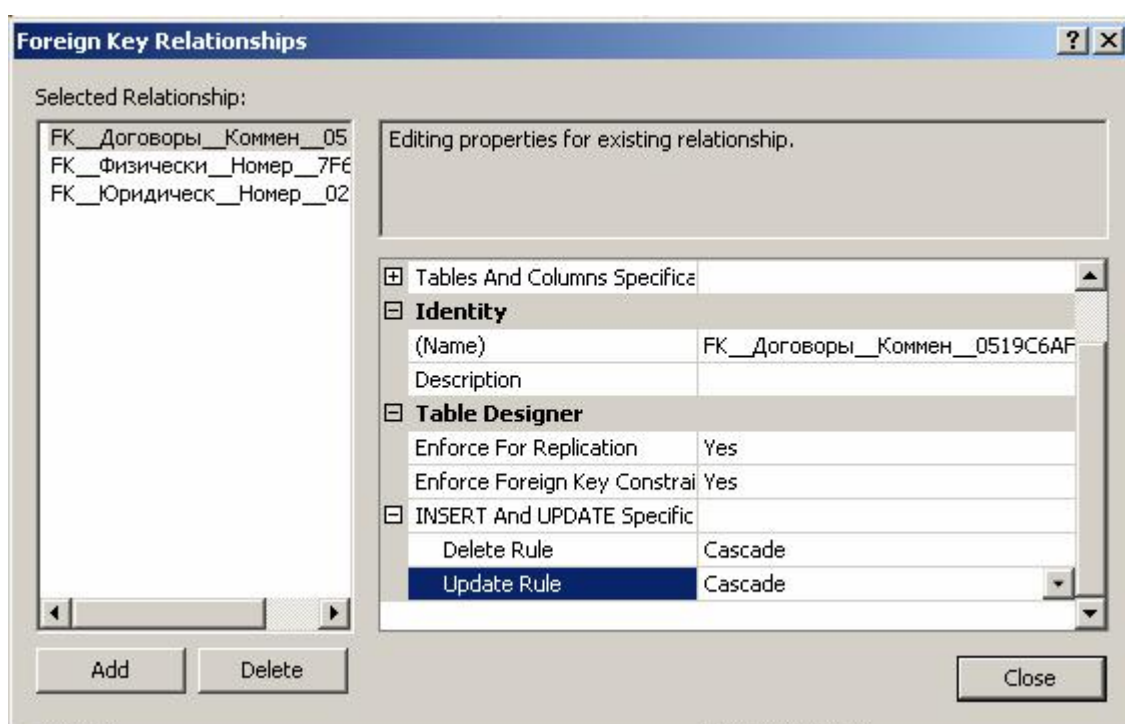


Рисунок 6.5

2. Предположим, что в силу каких-то причин возникла необходимость для поставщика с кодом 2 изменить код на 8. Выбрав соответствующую запись в таблице Поставщики, измените код поставщика с 2 на 8. Затем попытайтесь перейти на предыдущую запись. Проверьте изменения значения код поставщика в непосредственно связанных с этим поставщиком таблицах (ЮридическиеЛица, Договоры). Если изменения не появились сразу, то таблицу нужно закрыть и затем снова открыть или в окне содержимого таблицы щелкнуть правой кнопкой мыши и в появившемся меню выбрать пункт Execute SQL.
3. Теперь предположим, что данного поставщика (который теперь имеет код 8), необходимо удалить. Выбрав соответствующую запись в таблице Поставщики, нажмите правую кнопку мыши и в меню выберите пункт

Delete. Затем подтвердите удаление записи. После этого проверьте состояние данных в таблицах, которые прямо или косвенно связаны с данным поставщиком (ЮридическиеЛица, Договоры, Поставлено). Убедитесь в том, что соответствующие данные удалены. После этого нужно таблицы закрыть

III. Изучение особенностей работы механизма ссылочной целостности Set Null

Рассмотрим особенности работы механизма ссылочной целостности Cascade на примере отношений между таблицами Поставщики и Договоры. Доступ к связям выполняется также, как описано выше. Для изучения особенностей работы механизма ссылочной целостности выполним следующую последовательность действий.

1. В списке таблиц выбрать таблицу Поставщики, щелкнув по ней правой кнопкой мыши. В появившемся меню выбрать пункт Modify. В результате будет получен доступ к редактированию структуры таблицы. Для поля КодПоставщика установить свойство Allow Nulls (рисунок 6.6).

	Column Name	Data Type	Allow Nulls
►	НомерДоговора	int	<input type="checkbox"/>
	ДатаДоговора	datetime	<input checked="" type="checkbox"/>
	КодПоставщика	int	<input checked="" type="checkbox"/>
	Комментарий	text	<input checked="" type="checkbox"/>
			<input type="checkbox"/>

Рисунок 6.6

2. Щелкнуть правой кнопкой мыши по любому полю таблицы и в появившемся меню выбрать пункт Relationships... В результате на экране появится окно Foreign Key Relationships. Изменить механизмы ссылочной целостности для связи между всеми таблицами Поставщики и Договоры на Set Null (рисунок 6.7). Сохранить изменения в таблице.
3. Открыть в режиме просмотра данных таблицы Поставщики и Договоры. Для договора 6 изменить код поставщика с 1 на 7. Затем в таблице Поставщики изменить код поставщика 7 на 10. Проверить данные в таблице Договоры. Код поставщика в договоре 6 должен принять значение Null. Пример таблицы с измененными данными приведен на рисунке 6.8.
4. В таблице Договоры для договора 6 изменить значение код поставщика с Null на 10. После этого в таблице Поставщика удалить поставщика с кодом 10. Проверьте состояние данных в таблице Договоры. Для договора 6 значение код поставщика опять должно принять значение Null.

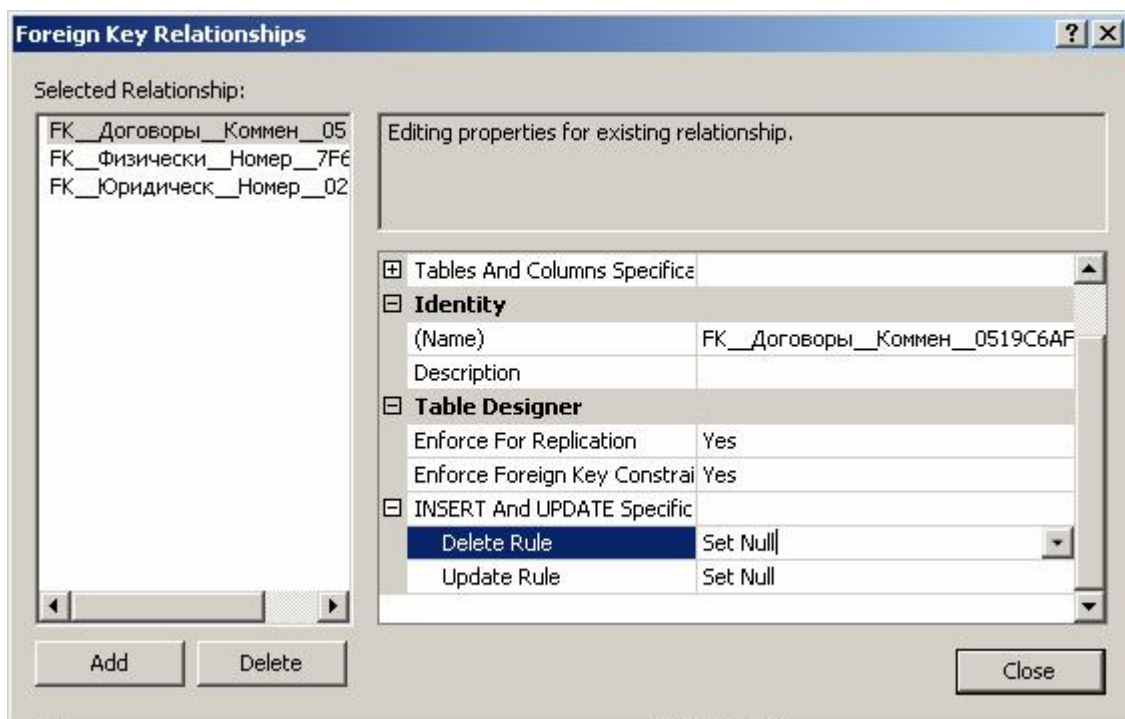


Рисунок 6.7

	НомерДоговора	ДатаДоговора	КодПоставщика	Комментарий
▶	1	01.09.1999 0:0...	1	Основание - на...
	2	10.09.1999 0:0...	1	Основание - сч...
	3	10.09.1999 0:0...	3	Основание - сч...
	4	23.09.1999 0:0...	3	Основание - за...
	6	01.10.1999 0:0...	NULL	Основание - сч...
*	NULL	NULL	NULL	NULL

Рисунок 6.8

5. Открытые для просмотра данных таблицы закрыть.

IV. Изучение особенностей работы механизма ссылочной целостности Set Default

Рассмотрим особенности работы механизма ссылочной целостности Set Default на примере отношений между таблицами Поставщики и Договоры. Доступ к связям выполняется также, как описано выше. Для изучения особенностей работы механизма ссылочной целостности выполним следующую последовательность действий.

1. Проверить наличие связи между таблицами Поставщики и Договоры. Это можно сделать, в частности, путем создания диаграммы базы данных. При отсутствии связи связь установить. Диаграмму закрыть и сохранить.
2. Открыть таблицу Поставщики в режиме просмотра данных. Для договора 6 изменить значение кода поставщика с Null на 3. Закрыть таблицу.

3. Открыть таблицу Поставщики в режиме редактирования структуры. Для поля КодПоставщика отключить свойство Allow Nulls (рисунок 6.9).



	Column Name	Data Type	Allow Nulls
	НомерДоговора	int	<input type="checkbox"/>
	ДатаДоговора	datetime	<input checked="" type="checkbox"/>
	КодПоставщика	int	<input type="checkbox"/>
	Комментарий	text	<input checked="" type="checkbox"/>
			<input type="checkbox"/>

Рисунок 6.9

4. Выбрать поле КодПоставщика и установить значение по умолчанию для этого поля. Для этого установить для свойства Default Value or Binding значение 1 (рисунок 6.10).

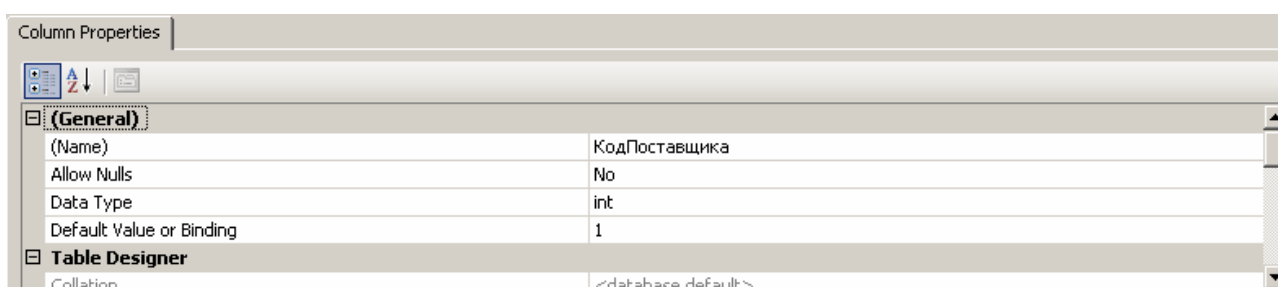


Рисунок 6.10

5. Щелкнуть правой кнопкой мыши по любому полю таблицы и в появившемся меню выбрать пункт Relationships.... В результате на экране появится окно Foreign Key Relationships. Изменить механизмы ссылочной целостности для связи между таблицами Поставщики и Договоры на Set Default (рисунок 6.11). Закрыть окно Foreign Key Relationships и сохранить изменения в таблице.

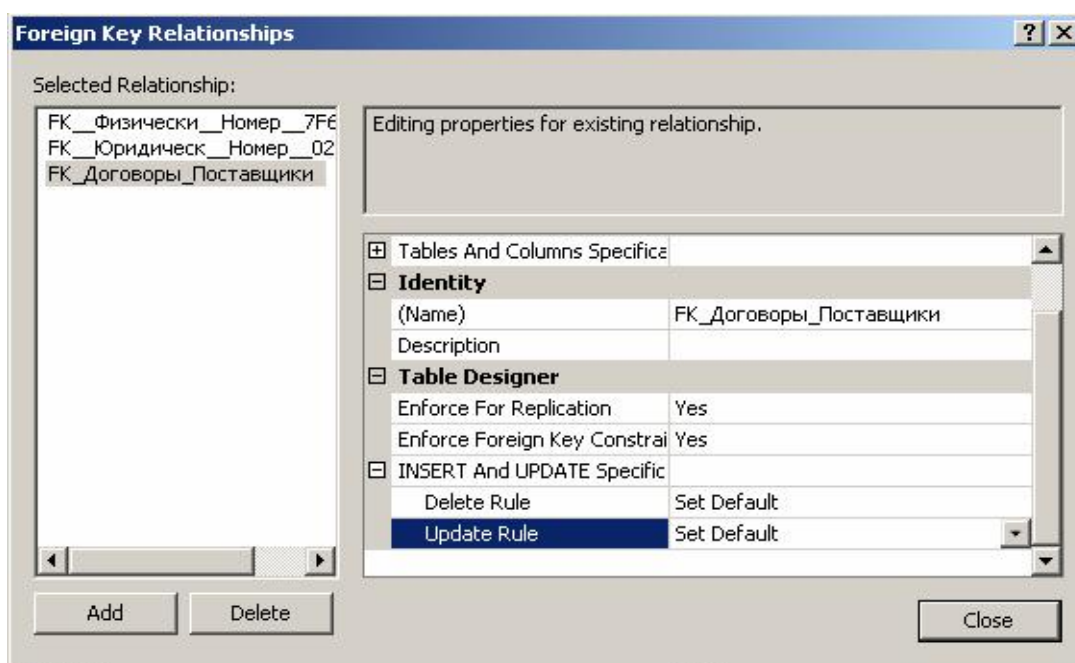


Рисунок 6.11

6. Открыть в режиме просмотра данных таблицы Поставщики и Договоры. В таблице Договоры определить список договоров, для которых код поставщика равен 3. В таблице Поставщики изменить код поставщика 3 на 12. Проверить данные в таблице Договоры. Для договоров, для которых код поставщика был равен 3, код поставщика должен измениться на 1.
7. В таблице Договоры изменить для некоторых договоров (например, для договоров 3, 4, 6) код поставщика с 1 на 12.
8. В таблице Поставщики удалить запись, соответствующую поставщику с кодом 12.
9. Проверить данные в таблице Договоры. Для договоров, для которых код поставщика был равен 12, код поставщика должен измениться на 1.

После окончания работы все таблицы нужно закрыть, а затем базу данных отключить. Базу данных можно не сохранять.

Требования к отчету:

- 1) кратко описать основные этапы выполнения задания;
- 2) описать особенности рассмотренных механизмов контроля ссылочной целостности и результаты их использования;
- 3) проанализировать, в каких ситуациях какие механизмы контроля ссылочной целостности предпочтительнее использовать и почему

Лабораторная работа 7

Использование транзакций

Перед началом выполнения работы рассмотрим краткий обзор основных положений, связанных с понятием транзакции и особенностями использования транзакционных механизмов.

Транзакция – это набор операций, который выполняется как один логический блок. Использование транзакций позволяет SQL Server обеспечивать определенный уровень целостности и восстанавливаемости данных. Журнал транзакций, который должна иметь каждая база данных, поддерживает запись всех транзакций, которые осуществляют любой тип модификации в базе данных (вставка, обновление или удаление). SQL Server использует этот журнал транзакций для восстановления данных в случае ошибок или отказов системы.

Целостность транзакции зависит, в частности, от программиста SQL. Программист должен знать, когда начинать и когда заканчивать транзакцию и в какой последовательности, чтобы модификации данных обеспечивали логическую согласованность и содержательность данных.

Чтобы транзакцию можно было считать допустимой для использования, она должна отвечать четырем требованиям. Эти требования называют ACID-свойствами. ACID – это сокращение от atomicity (атомарность), consistency (согласованность), isolation (изолированность) и durability (устойчивость). В SQL Server включены механизмы, помогающие обеспечивать соответствие транзакций каждому из этих требований.

Для запуска транзакции используется оператор T-SQL BEGIN TRANSACTION. Чтобы указать конец транзакции, используется COMMIT TRANSACTION или ROLLBACK TRANSACTION. В операторе BEGIN TRANSACTION вы можете дополнительно указать имя транзакции и затем сослаться на эту транзакцию по имени в операторе COMMIT TRANSACTION или ROLLBACK TRANSACTION. Ниже показан синтаксис этих трех операторов:

```
BEGIN TRAN[SACTION] [имя_транзакции | @переменная_с_именем_транзакции]
```

```
COMMIT [TRAN[SACTION] [имя_транзакции | @переменная_с_именем_транзакции]]
```

```
ROLLBACK [TRAN[SACTION] [имя_транзакции | @переменная_с_именем_транзакции  
| имя_точки_сохранения | @переменная_с_именем_точки_сохранения]]
```

В SQL Server разрешаются вложенные транзакции, т.е. транзакции внутри транзакции. В случае вложенных транзакций нужно явно фиксировать каждую внутреннюю транзакцию, чтобы SQL Server получал информацию об окончании внутренней транзакции и мог освободить ресурсы, используемые

этой транзакцией, когда будет фиксирована внешняя транзакция. Если ресурсы заблокированы, другие пользователи не могут получать доступа к этим ресурсам. Хотя нужно явным образом включать оператор фиксации COMMIT для каждой транзакции, SQL Server не выполняет фактического фиксирования внутренних транзакций, пока не произойдет успешное фиксирование внешней транзакции; одновременно с этим SQL Server освобождает все ресурсы, используемые внутренними и внешней транзакциями. При неуспешном фиксировании внешней транзакции фиксирование внутренних транзакций не выполняется и происходит откат внешней и всех внутренних транзакций. После фиксирования внешней транзакции выполняется фиксирование внутренних транзакций. Иными словами, SQL Server по сути игнорирует операторы COMMIT внутри внутренних вложенных транзакций – в том смысле, что внутренние транзакции не фиксируются в ожидании окончательного фиксирования или отката внешней транзакции, чтобы определить статус завершения всех внутренних транзакций. Кроме того, в случае использования оператора отката ROLLBACK во внешней транзакции или в любой из внутренних транзакций происходит откат всех этих транзакций. В оператор ROLLBACK нельзя включать имя внутренней транзакции; в этом случае SQL Server возвратит сообщение об ошибке. Можно включать имя внешней транзакции, имя точки сохранения или не включать никакого имени.

Хотя SQL Server не выполняет фактического фиксирования внутренних транзакций по оператору COMMIT, но все же для каждого оператора COMMIT происходит изменение системной переменной @@TRANCOUNT. Эта переменная следит за количеством активных транзакций на одно соединение с пользователем. При отсутствии активных транзакций значение @@TRANCOUNT равно 0. В начале каждой транзакции (с помощью BEGIN TRAN) значение @@TRANCOUNT увеличивается на 1. После фиксирования каждой транзакции значение @@TRANCOUNT уменьшается на 1. Когда значение @@TRANCOUNT становится равным 0, фиксируется внешняя транзакция. Если во внешней транзакции или в любой из внутренних транзакций выполняется оператор ROLLBACK, то значение @@TRANCOUNT устанавливается равным 0. Помните, что для правильного уменьшения @@TRANCOUNT вы должны указывать фиксирование (COMMIT) для каждой внутренней транзакции. Вы можете проверять значение @@TRANCOUNT, чтобы определять наличие активных транзакций. Чтобы увидеть значение @@TRANCOUNT, используйте оператор SELECT @@TRANCOUNT.

В неявном режиме транзакция автоматически начинается при использовании определенных операторов T-SQL и продолжается, пока не появится оператор явного окончания COMMIT или ROLLBACK. Если оператор окончания не указан, то при отсоединении пользователя происходит откат данной транзакции. Следующие операторы T-SQL являются началом новой транзакции в неявном режиме:

ALTER TABLE
CREATE
DELETE
DROP
FETCH
GRANT
INSERT
OPEN
REVOKE
SELECT
TRUNCATE TABLE
UPDATE

Если один из этих операторов используется, чтобы начать неявную транзакцию, эта транзакция продолжается, пока не будет явно указано ее окончание, даже если внутри транзакции снова встретятся эти операторы. После явного фиксирования или отката данной транзакции следующее появление этих операторов является началом новой транзакции. Этот процесс продолжает действовать, пока не будет отключен неявный режим.

Чтобы задать неявный режим транзакций, вы можете использовать следующий оператор T-SQL:

```
SET IMPLICIT_TRANSACTIONS {ON | OFF}
```

Значение ON активизирует неявный режим, и OFF отключает его. После отключения неявного режима используется режим автофиксации.

Неявные транзакции полезно использовать, когда запускаются сценарии с модификациями данных, которые требуется защитить внутри транзакции. Вы можете включить неявный режим в начале сценария, выполнить необходимые модификации и отключить этот режим в конце сценария. Во избежание конфликтов параллельных операций отключайте неявный режим после модификации данных и перед просмотром данных. Если вслед за операцией фиксирования следует оператор SELECT, то с него начинается новая транзакция в неявном режиме, и соответствующие ресурсы не будут освобождены, пока не будет фиксирована эта транзакция.

Откаты транзакций могут происходить в двух формах: автоматический откат, выполняемый SQL Server, или программируемый вручную откат. В определенных случаях SQL Server выполнит для вас откат. Но для обеспечения логической согласованности в программах нужно при необходимости явным образом обращаться к оператору ROLLBACK. Рассмотрим более подробно эти два метода.

При сбое транзакции вследствие серьезной ошибки, такой как потеря сетевого соединения при выполнении данной транзакции или отказ клиентского приложения или компьютера, SQL Server выполняет автоматический откат транзакции. Во время отката происходит отмена всех

модификаций, выполненных в данной транзакции, и освобождение всех ресурсов, использовавшихся этой транзакцией. Если при исполнении какого-либо оператора возникает ошибка, такая как нарушение ограничения или правила, то по умолчанию SQL Server автоматически выполняет откат только этого определенного оператора, в котором возникла ошибка.

При программируемых откатах с помощью оператора ROLLBACK вы можете указать точку в транзакции, где будет выполнен откат. Оператор ROLLBACK прекращает данную транзакцию и выполняет откат (отмену) всех выполненных изменений. Если вы запускаете откат в середине какой-либо транзакции, то остальная часть этой транзакции игнорируется. Если, например, эта транзакция является хранимой процедурой и оператор ROLLBACK выполняется в этой процедуре, то происходит откат всей процедуры и переход к обработке оператора, следующего после вызова хранимой процедуры.

Откат транзакции нельзя выполнить после ее фиксирования. Чтобы можно было выполнить явный откат отдельной транзакции, оператор ROLLBACK должен предшествовать оператору COMMIT. В случае вложенных транзакций после фиксирования внешней транзакции (и, тем самым, внутренних транзакций) уже нельзя выполнить откат ни одной из транзакций. Как уже отмечалось, вы не можете выполнить откат только внутренних транзакций; должен быть выполнен откат всей транзакции (всех внутренних транзакций и внешней транзакции). Поэтому, включив имя транзакции в оператор ROLLBACK, проследите за тем, чтобы было указано имя внешней транзакции (во избежание путаницы и сообщения об ошибке от SQL Server). Однако существует обходной путь, позволяющий избежать отката всей транзакции и сохранить часть модификаций: вы можете использовать точки сохранения, которые позволяют выполнять откат только до определенной точки в транзакции, а не до самого начала транзакции. Все модификации, выполненные до точки сохранения, остаются в силе и не подвергаются откату; но для всех операторов, выполняемых после точки сохранения (которую вы должны указать в транзакции) вплоть до оператора ROLLBACK, будет выполнен откат. Затем начнут выполняться операторы, следующие за оператором ROLLBACK. Если вы затем зададите откат транзакции без указания точки сохранения, то, как обычно, будет выполнен откат всех модификаций вплоть до начала данной транзакции, т.е. будет отменена вся транзакция. Отметим, что при откате транзакции до точки сохранения SQL Server не освобождает заблокированные ресурсы. Они будут освобождены после фиксирования транзакции или после отката всей транзакции.

Чтобы указать точку сохранения в транзакции, используйте следующий оператор:

```
SAVE TRAN[SACTION] {имя_точки_сохранения |  
@переменная_с_именем_точки_сохранения}
```

Поместите точку сохранения в том месте транзакции, до которого вы хотите выполнять откат. Чтобы выполнить откат до точки сохранения,

используйте оператор **ROLLBACK TRAN** вместе с именем точки сохранения, как это показано ниже:

```
ROLLBACK TRAN имя_точки_сохранения
```

Блокировка транзакций.

В SQL Server используется объект, который называется блокировкой (lock); он препятствует тому, чтобы несколько пользователей одновременно вносили изменения в базу данных и чтобы один пользователь считывал данные, которые изменяет в этот момент другой пользователь. Блокировка помогает обеспечивать логическую целостность транзакций и данных. Управление блокировками осуществляется внутренним образом из программного обеспечения SQL Server и захват блокировки осуществляется на уровне пользовательского соединения. Если пользователь захватывает блокировку (становится ее владельцем) по какому-либо ресурсу, то эта блокировка указывает, что данный пользователь имеет право на использование данного ресурса. К ресурсам, которые может блокировать пользователь, относятся строка данных, страница данных экстенст (8 страниц), таблица или вся база данных. Например, если пользователь владеет блокировкой по странице данных, то другой пользователь не может выполнять операции на этой странице, которые повлияют на операции пользователя, владеющего данной блокировкой. Поэтому любой пользователь не может модифицировать страницу данных, которая заблокирована и считывается в данный момент другим пользователем. Кроме того, ни один пользователь не может владеть блокировкой, конфликтующей с блокировкой, которой уже владеет другой пользователь. Например, два пользователя не могут одновременно владеть блокировками на одновременную модификацию одной и той же страницы. Одна блокировка не может одновременно использоваться более чем одним пользователем.

ВЫПОЛНЕНИЕ РАБОТЫ

Внимание! Как и при выполнении предыдущей работы, рекомендуется использовать отдельную базу данных. С учетом этой особенности в качестве имени базы данных в текстах запросов может использоваться не `delivery`, а какое-то другое.

I. Создать запрос, иллюстрирующий работу транзакционного механизма при добавлении данных в одну таблицу

Рассмотрим последовательность действий при создании и использовании запроса, с помощью которого запускается транзакция, в таблицу `Поставлено` добавляется новая запись, а затем имитируется ситуация некорректного или корректного завершения транзакции. Состояние таблицы контролируется до начала транзакции, в процессе выполнения транзакции и после завершения транзакции. Для этого нужно выполнить следующую последовательность действий.

1. На панели инструментов нажать кнопку `New Query`
2. Ввести текст запроса, приведенный на рисунке 7.1

```
USE delivery

SELECT Поставлено.НомерДоговора, Поставлено.Товар, Поставлено.Цена, Поставлено.Количество,
       Поставщики.*, Договоры.ДатаДоговора
FROM Поставлено, Договоры, Поставщики
WHERE Договоры.НомерДоговора = Поставлено.НомерДоговора
AND Поставщики.КодПоставщика = Договоры.КодПоставщика AND Договоры.НомерДоговора = 1

BEGIN TRANSACTION
INSERT INTO Поставлено VALUES (1, 'Пылесос', 22, 389.75);

SELECT Поставлено.НомерДоговора, Поставлено.Товар, Поставлено.Цена, Поставлено.Количество,
       Поставщики.*, Договоры.ДатаДоговора
FROM Поставлено, Договоры, Поставщики
WHERE Договоры.НомерДоговора = Поставлено.НомерДоговора
AND Поставщики.КодПоставщика = Договоры.КодПоставщика AND Договоры.НомерДоговора = 1

ROLLBACK

SELECT Поставлено.НомерДоговора, Поставлено.Товар, Поставлено.Цена, Поставлено.Количество,
       Поставщики.*, Договоры.ДатаДоговора
FROM Поставлено, Договоры, Поставщики
WHERE Договоры.НомерДоговора = Поставлено.НомерДоговора
AND Поставщики.КодПоставщика = Договоры.КодПоставщика AND Договоры.НомерДоговора = 1
```

Рисунок 7.1

3. Выполнить запрос.
4. В случае успешного выполнения запроса на экран будут выведены данные, иллюстрирующие состояние таблицы до начала транзакции, в процессе выполнения транзакции и после завершения транзакции (рисунок 7.2). Как видно из приведенных данных, новая запись в таблице появляется, а затем исчезает

Results Messages

	НомерДогово...	Товар	Цена	Количество	КодПоставщи...	Адрес	Примечание	ДатаДоговора
1	1	Видеомагнитофон	722.33	12	1	г.Харьков, пр. Ленина, 55, к.108	тел. 32-18-44	1999-09-01 00:00
2	1	Компьютер	1554.22	24	1	г.Харьков, пр. Ленина, 55, к.108	тел. 32-18-44	1999-09-01 00:00
3	1	Магнитофон	655.12	25	1	г.Харьков, пр. Ленина, 55, к.108	тел. 32-18-44	1999-09-01 00:00
4	1	Стереосистема	220.45	12	1	г.Харьков, пр. Ленина, 55, к.108	тел. 32-18-44	1999-09-01 00:00
5	1	Телевизор	1253.45	10	1	г.Харьков, пр. Ленина, 55, к.108	тел. 32-18-44	1999-09-01 00:00

	НомерДогово...	Товар	Цена	Количество	КодПоставщи...	Адрес	Примечание	ДатаДоговора
1	1	Видеомагнитофон	722.33	12	1	г.Харьков, пр. Ленина, 55, к.108	тел. 32-18-44	1999-09-01 00:00
2	1	Компьютер	1554.22	24	1	г.Харьков, пр. Ленина, 55, к.108	тел. 32-18-44	1999-09-01 00:00
3	1	Магнитофон	655.12	25	1	г.Харьков, пр. Ленина, 55, к.108	тел. 32-18-44	1999-09-01 00:00
4	1	Пылесос	389.75	22	1	г.Харьков, пр. Ленина, 55, к.108	тел. 32-18-44	1999-09-01 00:00
5	1	Стереосистема	220.45	12	1	г.Харьков, пр. Ленина, 55, к.108	тел. 32-18-44	1999-09-01 00:00
6	1	Телевизор	1253.45	10	1	г.Харьков, пр. Ленина, 55, к.108	тел. 32-18-44	1999-09-01 00:00

	НомерДогово...	Товар	Цена	Количество	КодПоставщи...	Адрес	Примечание	ДатаДоговора
1	1	Видеомагнитофон	722.33	12	1	г.Харьков, пр. Ленина, 55, к.108	тел. 32-18-44	1999-09-01 00:00
2	1	Компьютер	1554.22	24	1	г.Харьков, пр. Ленина, 55, к.108	тел. 32-18-44	1999-09-01 00:00
3	1	Магнитофон	655.12	25	1	г.Харьков, пр. Ленина, 55, к.108	тел. 32-18-44	1999-09-01 00:00
4	1	Стереосистема	220.45	12	1	г.Харьков, пр. Ленина, 55, к.108	тел. 32-18-44	1999-09-01 00:00
5	1	Телевизор	1253.45	10	1	г.Харьков, пр. Ленина, 55, к.108	тел. 32-18-44	1999-09-01 00:00

Рисунок 7.2

5. Теперь рассмотрим ситуацию корректного завершения транзакции. Для этого в приведенном тексте запроса изменим оператор ROLLBACK на COMMIT. Выполним запрос. Результат приведен на рисунке 7.3. Запрос можно сохранить с именем SQLQuery_trans.sql

Results Messages

	НомерДогово...	Товар	Цена	Количество	КодПоставщи...	Адрес	Примечание	ДатаДоговора
1	1	Видеомагнитофон	722.33	12	1	г.Харьков, пр. Ленина, 55, к.108	тел. 32-18-44	1999-09-01 00:00
2	1	Компьютер	1554.22	24	1	г.Харьков, пр. Ленина, 55, к.108	тел. 32-18-44	1999-09-01 00:00
3	1	Магнитофон	655.12	25	1	г.Харьков, пр. Ленина, 55, к.108	тел. 32-18-44	1999-09-01 00:00
4	1	Стереосистема	220.45	12	1	г.Харьков, пр. Ленина, 55, к.108	тел. 32-18-44	1999-09-01 00:00
5	1	Телевизор	1253.45	10	1	г.Харьков, пр. Ленина, 55, к.108	тел. 32-18-44	1999-09-01 00:00

	НомерДогово...	Товар	Цена	Количество	КодПоставщи...	Адрес	Примечание	ДатаДоговора
1	1	Видеомагнитофон	722.33	12	1	г.Харьков, пр. Ленина, 55, к.108	тел. 32-18-44	1999-09-01 00:00
2	1	Компьютер	1554.22	24	1	г.Харьков, пр. Ленина, 55, к.108	тел. 32-18-44	1999-09-01 00:00
3	1	Магнитофон	655.12	25	1	г.Харьков, пр. Ленина, 55, к.108	тел. 32-18-44	1999-09-01 00:00
4	1	Пылесос	389.75	22	1	г.Харьков, пр. Ленина, 55, к.108	тел. 32-18-44	1999-09-01 00:00
5	1	Стереосистема	220.45	12	1	г.Харьков, пр. Ленина, 55, к.108	тел. 32-18-44	1999-09-01 00:00
6	1	Телевизор	1253.45	10	1	г.Харьков, пр. Ленина, 55, к.108	тел. 32-18-44	1999-09-01 00:00

	НомерДогово...	Товар	Цена	Количество	КодПоставщи...	Адрес	Примечание	ДатаДоговора
1	1	Видеомагнитофон	722.33	12	1	г.Харьков, пр. Ленина, 55, к.108	тел. 32-18-44	1999-09-01 00:00
2	1	Компьютер	1554.22	24	1	г.Харьков, пр. Ленина, 55, к.108	тел. 32-18-44	1999-09-01 00:00
3	1	Магнитофон	655.12	25	1	г.Харьков, пр. Ленина, 55, к.108	тел. 32-18-44	1999-09-01 00:00
4	1	Пылесос	389.75	22	1	г.Харьков, пр. Ленина, 55, к.108	тел. 32-18-44	1999-09-01 00:00
5	1	Стереосистема	220.45	12	1	г.Харьков, пр. Ленина, 55, к.108	тел. 32-18-44	1999-09-01 00:00
6	1	Телевизор	1253.45	10	1	г.Харьков, пр. Ленина, 55, к.108	тел. 32-18-44	1999-09-01 00:00

Рисунок 7.3

II. Создать запрос, иллюстрирующий работу транзакционного механизма при добавлении данных в несколько таблиц

Рассмотрим последовательность действий при создании и использовании запроса, с помощью которого запускается транзакция, в затем создается новый поставщик, с этим поставщиком заключается договор на поставку, по этому договору поставляется продукция. Имитируется ситуация некорректного или корректного завершения транзакции. Состояние таблиц контролируется до начала транзакции, в процессе выполнения транзакции и после завершения транзакции. Для этого нужно выполнить следующую последовательность действий.

1. На панели инструментов нажать кнопку New Query
2. Ввести текст запроса, приведенный на рисунке 7.4

```
USE delivery

SELECT * FROM Поставщики
SELECT * FROM Договоры
SELECT * FROM Поставлено

BEGIN TRANSACTION
INSERT INTO Поставщики (КодПоставщика, Адрес, Примечание)
VALUES (6, 'г. Богодухов, ул. Маркса, 55', '');
INSERT INTO Договоры (ДатаДоговора, КодПоставщика, Комментарий)
VALUES ('20021212', 6, '');
INSERT INTO Поставлено VALUES (8, 'Пылесос', 22, 389.75);
INSERT INTO Поставлено VALUES (8, 'Кофемолка', 33, 89.45);

SELECT * FROM Поставщики
SELECT * FROM Договоры
SELECT * FROM Поставлено

ROLLBACK

SELECT * FROM Поставщики
SELECT * FROM Договоры
SELECT * FROM Поставлено
```

Рисунок 7.4

3. Выполнить запрос.
4. В случае успешного выполнения запроса на экран будут выведены данные, иллюстрирующие состояние таблиц до начала транзакции, в процессе выполнения транзакции и после завершения транзакции (аналогично предыдущему запросу). Как видно из приведенных данных, новые записи в таблицах появляются, а затем исчезают
5. Теперь рассмотрим ситуацию корректного завершения транзакции. Для этого в приведенном тексте запроса изменим оператор ROLLBACK на COMMIT. Выполним запрос. В результате выполнения запроса данные

должны быть внесены в таблицы и сохранены. В этом нужно убедиться, открыв соответствующие таблицы в режиме просмотра данных. Запрос можно сохранить с именем SQLQuery_trans1.sql

III. Создать запрос, иллюстрирующий работу транзакционного механизма при изменении данных в нескольких таблицах

Рассмотрим последовательность действий при создании и использовании запроса, с помощью которого запускается транзакция, затем изменяются данные, введенные в таблицы при выполнении предыдущего запроса. Имитируется ситуация некорректного или корректного завершения транзакции. Состояние таблиц контролируется до начала транзакции, в процессе выполнения транзакции и после завершения транзакции. Для этого нужно выполнить следующую последовательность действий.

1. Для отношений ссылочной целостности между всеми таблицами базы данных установить механизм Cascade.
2. На панели инструментов нажать кнопку New Query
3. Ввести текст запроса, приведенный на рисунке 7.5

```
USE delivery

SELECT * FROM Поставщики
SELECT * FROM Договоры
SELECT * FROM Поставлено WHERE НомерДоговора=8

BEGIN TRANSACTION
UPDATE Поставщики SET КодПоставщика = 22 WHERE КодПоставщика = 6
UPDATE Поставлено SET Цена = Цена * 1.1 WHERE НомерДоговора = 8

SELECT * FROM Поставщики
SELECT * FROM Договоры
SELECT * FROM Поставлено WHERE НомерДоговора=8

ROLLBACK

SELECT * FROM Поставщики
SELECT * FROM Договоры
SELECT * FROM Поставлено WHERE НомерДоговора=8
```

Рисунок 7.5

4. Выполнить запрос.
5. В случае успешного выполнения запроса на экран будут выведены данные, иллюстрирующие состояние таблиц до начала транзакции, в процессе выполнения транзакции и после завершения транзакции (аналогично предыдущим запросам). Как видно из приведенных данных, изменения данных в таблицах появляются, а затем исчезают

6. Теперь рассмотрим ситуацию корректного завершения транзакции. Для этого в приведенном тексте запроса изменим оператор ROLLBACK на COMMIT. Выполним запрос. В результате выполнения запроса данные должны быть внесены в таблицы и сохранены. В этом нужно убедиться, открыв соответствующие таблицы в режиме просмотра данных. Запрос можно сохранить с именем SQLQuery_trans2.sql

IV. Создать запрос, иллюстрирующий работу транзакционного механизма при удалении данных в несколько таблиц

Рассмотрим последовательность действий при создании и использовании запроса, с помощью которого запускается транзакция, в рамках которой удаляется поставщик, который был создан при выполнении запроса II и данные которого были изменены при выполнении запроса III. С учетом используемого механизма контроля ссылочной целостности (Cascade) данные будут удалены в нескольких таблицах. Имитируется ситуация некорректного или корректного завершения транзакции. Состояние таблиц контролируется до начала транзакции, в процессе выполнения транзакции и после завершения транзакции. Для этого нужно выполнить следующую последовательность действий.

1. На панели инструментов нажать кнопку New Query
2. Ввести текст запроса, приведенный на рисунке 7.6

```
USE delivery

SELECT * FROM Поставщики
SELECT * FROM Договоры
SELECT * FROM Поставлено

BEGIN TRANSACTION
DELETE FROM Поставщики WHERE КодПоставщика = 6

SELECT * FROM Поставщики
SELECT * FROM Договоры
SELECT * FROM Поставлено

ROLLBACK

SELECT * FROM Поставщики
SELECT * FROM Договоры
SELECT * FROM Поставлено
```

Рисунок 7.6

3. Выполнить запрос.
4. В случае успешного выполнения запроса на экран будут выведены данные, иллюстрирующие состояние таблиц до начала транзакции, в процессе

выполнения транзакции и после завершения транзакции (аналогично предыдущим запросам). Как видно из приведенных данных, изменения данных в таблицах появляются, а затем исчезают

5. Теперь рассмотрим ситуацию корректного завершения транзакции. Для этого в приведенном тексте запроса изменим оператор ROLLBACK на COMMIT. Выполним запрос. В результате выполнения запроса данные должны быть внесены в таблицы и сохранены. В этом нужно убедиться, открыв соответствующие таблицы в режиме просмотра данных. Запрос можно сохранить с именем SQLQuery_trans3.sql

Сохранение результатов работы

Сохранить файлы запросов:

SQLQuery_trans.sql; SQLQuery_trans1.sql;

SQLQuery_trans2.sql; SQLQuery_trans3.sql

Требования к отчету:

- 1) кратко описать основные этапы выполнения задания;
- 2) описать особенности транзакционных механизмов контроля и результаты их использования;

Лабораторная работа 8

Использование СУБД MS Access как средства создания клиентского приложения

I. Создание ODBC-источника данных

Запуск средств доступа к ODBC-источникам данных

1. Открыть Панель управления (Control Panel) Windows
2. Открыть папку Администрирование (Administrative Tools)
3. Выбрать пункт Источники данных (ODBC) (Data Sources (ODBC))
4. Открыть список ODBC-источников двойным щелчком мыши

Создать новый ODBC-источник, для чего:

1. Выбрать вкладку Пользовательский DSN (User DSN) и нажать кнопку Добавить (Add)
2. Выбрать драйвер SQL Server и нажать кнопку Готово (Finish)
3. Ввести имя ODBC-источника и выбрать сервер, к которому нужно подключиться (рисунок 8.1). При выборе сервера нужно выбрать сервер, соответствующий компьютеру, на котором выполняется работа.

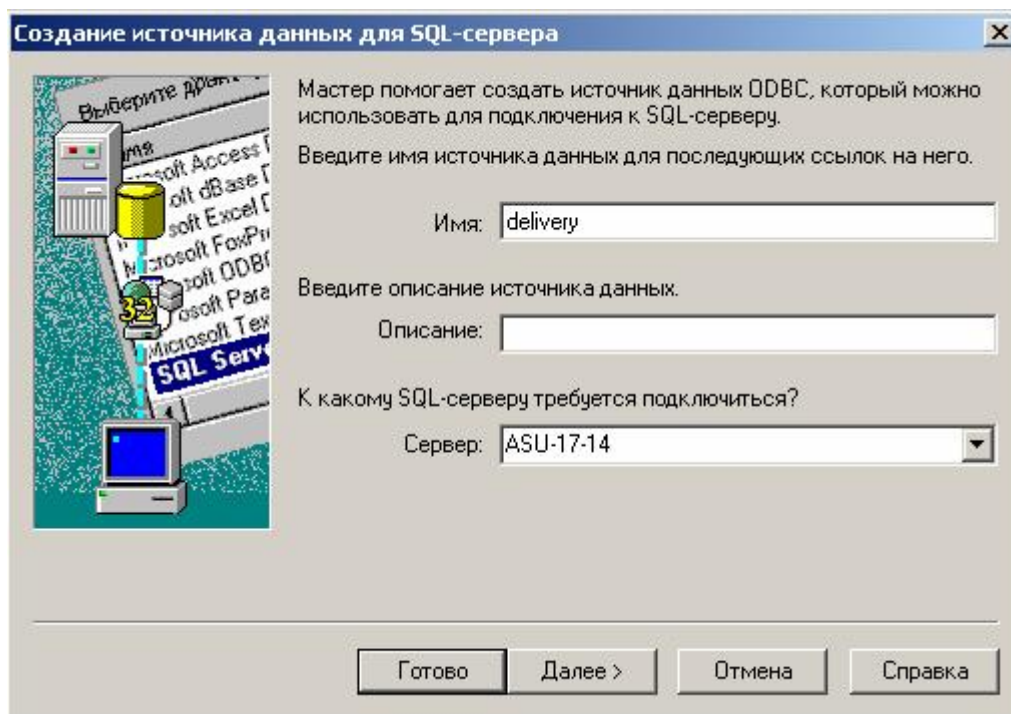


Рисунок 8.1

4. В окне проверки подлинности пользователя оставить данные без изменений (рисунок 8.2) и нажать кнопку Next (Далее)

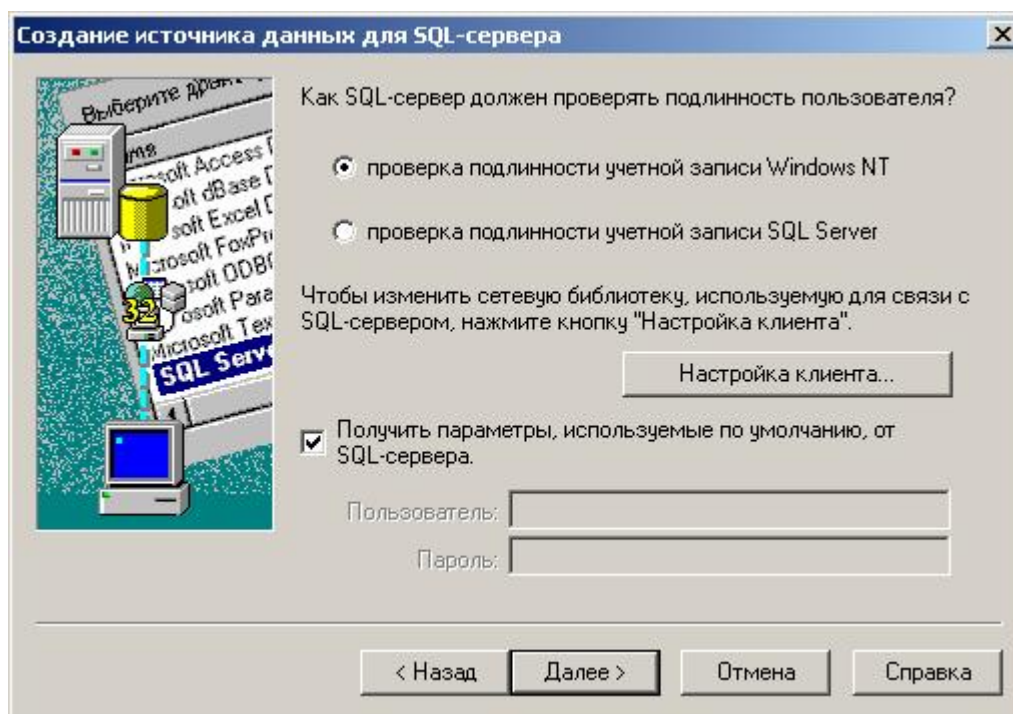


Рисунок 8.2

5. Выбрать базу данных, к которой по умолчанию будет осуществлено подключение (рисунок 8.3) и нажать кнопку Next (Далее). В следующем окне нажать кнопку Готово (Finish)

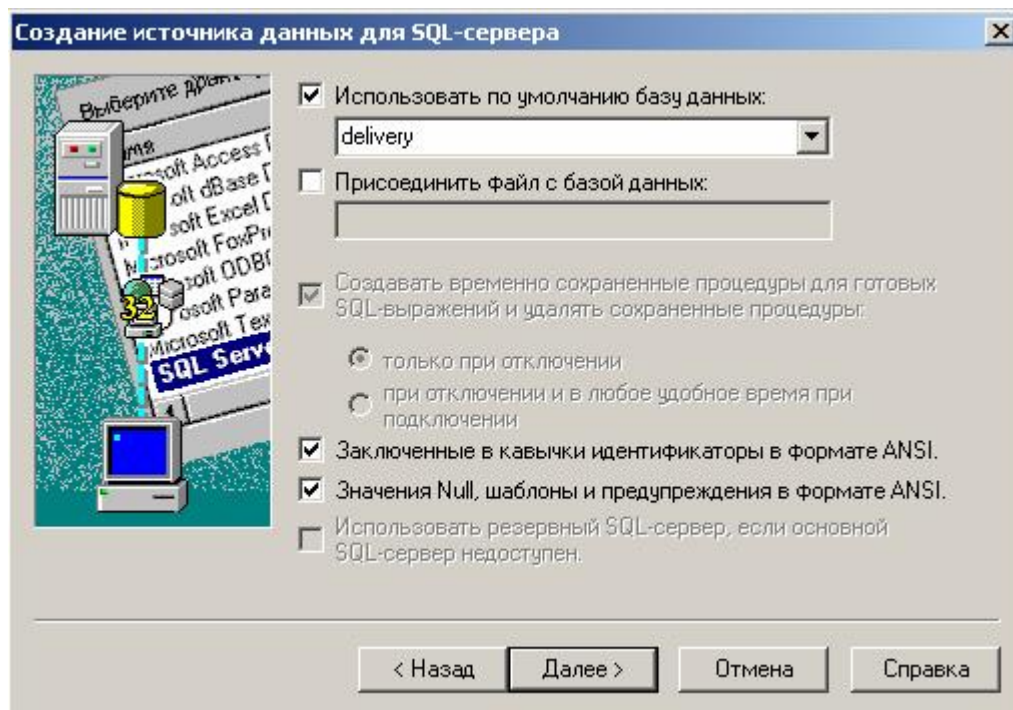


Рисунок 8.3

6. Проверить подключение к серверу и в случае успешного подключения нажать кнопку ОК. В этом случае ODBC-источник будет сохранен и появится в списке ODBC-источников

II. Использование СУБД MS Access в качестве клиентского приложения

Запустить СУБД MS Access

Создать новую базу данных с именем **client_mssql** (желательно в той же папке, где находится база данных, созданная при выполнении лабораторных работ)

В подменю пункта File выбрать пункт Внешние данные (Get External Data) и подпункт Связь с таблицами (Link Tables...) (рисунок 8.4)

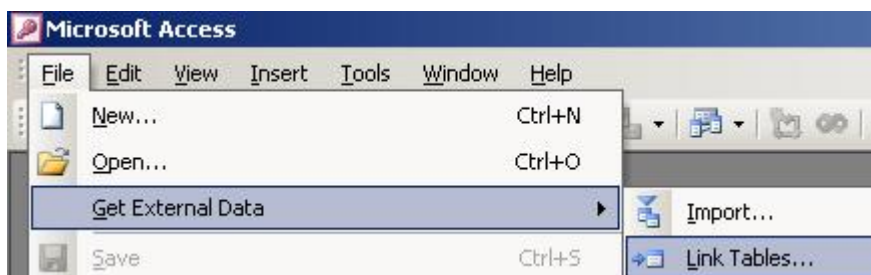


Рисунок 8.4

В окне Связь (Link), в поле со списком Тип файлов (Files of type) выбрать пункт Базы данных ODBC () (ODBC Databases ())

В окне Выбор источника данных (Select Data Source) открыть вкладку Источник данных компьютера (Machine Data Source), выбрать ODBC-источник с именем delivery

В окне Связь с таблицами (Link Tables) выбрать таблицу dbo.Поставщики. В результате будет создана таблица dbo_Поставщики

Аналогично создать таблицу dbo_Договоры, которая будет связана с таблицей Договоры.

Проверить возможность работы с базой данных, используя в качестве интерфейса пользователя СУБД MS Access. Для этого открыть созданные таблицы dbo_Поставщики (рисунок 8.5) и dbo_Договоры. Используя интерфейс MS Access, выполнить для проверки операции манипулирования данными (добавление, удаление, изменение данных). Проверить сохранение результатов манипулирования, используя SQL Server Management Studio

The image shows a screenshot of the Microsoft Access table view for the 'dbo_Поставщики' table. The table has three columns: 'КодПоставщика' (Supplier Code), 'Адрес' (Address), and 'Примечание' (Remarks). There are five rows of data listed. At the bottom, there is a status bar showing 'Запись: 1 из 6' (Record: 1 of 6).

КодПоставщика	Адрес	Примечание
1	г.Харьков, пр. Ленина, 55, к.108	тел. 32-18-44
2	г. Киев, пр. Победы, 154, к. 3	
3	г. Харьков, ул. Пушкинская, 77	тел.33-33-44, 12-34-56, факс 22-12-33
4	г. Одесса, ул. Дерибасовская, 75	
5	г. Полтава, ул. Ленина, 15, кв. 43	

Рисунок 8.5

Разработать средствами MS Access экранную форму, которая позволит для каждого поставщика видеть список заключенных с ним договоров. Возможный вариант реализации такой формы приведен на рисунке 8.6. Как видно из рисунка, форма состоит из главной и подчиненной форм.

НомерДоговора	ДатаДоговора	КодПоставщика	Комментарий
5	24.09.1999	2	Основание – накладная № 74 от 11/09/99
7	02.10.1999	2	Основание – накладная № 85 от 21/09/99
*(Счетчик)		2	

Рисунок 8.6

Эти главная и подчиненная формы в режиме конструктора приведены на рисунках 8.7 и 8.8.

Рисунок 8.7

The screenshot shows a Microsoft Access form titled "dbo_Договоры : форма". The form is divided into sections: "Заголовок формы" (Form Header) and "Область данных" (Data Area). The data area contains a table with the following fields: "НомерДоговора", "ДатаДогово", "КодПостав", and "Комментарий". The form is displayed in a window with a standard Windows interface, including a title bar, menu bar, and toolbar.

Рисунок 8.8

После создания формы проверить возможность работы с базой данных, используя в качестве интерфейса пользователя созданную форму.

Используя ранее полученные ранее навыки работы с СУБД MS Access создать другие формы, позволяющие работать с базой данных, а также средства обработки данных (запросы, отчеты), позволяющие обрабатывать информацию, хранящуюся в базе данных, выводить ее на печать и т.д.

Сохранение результатов работы

Сохранить файл client_mssql.mdb

Требования к отчету:

- 1) кратко описать основные этапы выполнения задания;
- 2) описать созданное клиентское приложение и особенности работы с ним