Министерство сельского хозяйства Российской Федерации
Федеральное государственное бюджетное образовательное учреждение высшего профессионального образования
"Кубанский государственный аграрный университет"
(КубГАУ)

Кафедра компьютерных технологий и систем

С.В. Лаптев

ИНФОРМАЦИОННЫЕ ТЕХНОЛОГИИ

Лабораторный практикум для студентов специальности
21.03.02 - "Землеустройство и кадастры"
(прикладной и академический бакалавриат)

Краснодар
2017

УДК 004.42 (073.8)
ББК 32.973.26-018
А-45

Рецензенты:
Барановская Т.П. – профессор, доктор экономических наук, заведующая кафедрой системного анализа и обработки информации Кубанского государственного аграрного университета (г. Краснодар);
Степанов В.В. – профессор, доктор технических наук, заведующий кафедрой вычислительной техники и автоматизированных систем (ВТиАС) Кубанского государственного технологического университета (г. Краснодар)

Информационные технологии: Лабораторный практикум для студентов специальности 21.03.02 "Землеустройство и кадастры" (прикладной и академический бакалавриат)- / С. В. Лаптев, – Краснодар: ФГОУ ВО КубГАУ, 2017. – 135 с.

Составлен в соответствии с рабочей программой дисциплины **Информационные технологии** для студентов второго курса специальности 21.03.02 – “ Землеустройство и кадастры ”.

Работа выполнена по решению методической комиссии факультета прикладной информатики и кафедры компьютерных технологий и систем (протокол №2 от 28.02.2017 г.)

Лабораторная работа №2.
Язык структурированных запросов (SQL) в СУБД Access.
Создание, редактирование и удаление таблиц.
Манипулирование данными в таблицах.

SQL (Structured Query Language) – стандартный язык, предназначенный для создания, модификации и управления данными в реляционных БД.

Основные категории команд языка SQL предназначены для выполнения различных функций, включая построение объектов базы данных и манипулирование ими, начальную загрузку данных в таблицы, обновление и удаление существующей информации, выполнение запросов к базе данных, управление доступом к ней и ее общее администрирование.

Для создания и изменения структуры объектов (таблиц) используется категория команд DDL (язык определения данных), для изменения данных внутри объекта (таблицы) – категория DML (язык манипулирования данными).

Для создания, редактирования и удаления таблиц используются операторы CREATE TABLE, ALTER TABLE и DROP TABLE соответственно.

Оператор CREATE TABLE имеет в общем случае следующее формальное описание:

CREATE [{ GLOBAL | LOCAL }] TEMPORARY] **TABLE** имя_таблицы
({ *column* | [table_constraint] } . , ..
[ON COMMIT { DELETE | PRESERVE} ROWS]);

column определяется как:

имя_поля {domain | datatype [size]}
[column_constraint…]
[DEFAULT default_value]
[COLLATE collate_value]

Фразы GLOBAL TEMPORARY или LOCAL TEMPORARY указывают на создание временной таблицы.

Фраза ON COMMIT может быть указана только для временных таблиц. По умолчанию для временных таблиц подразумевается фраза ON COMMIT DELETE ROWS.
После имени таблицы в круглых скобках через запятую указывается список полей (называемых также столбцами) и ограничений. Каждое поле имеет имя и тип (datatype). Тип может быть определен как любой допустимый тип языка SQL или как домен. Например, язык SQL допускает такие типы как: integer, char (число символов), varchar (число символов), int, smallint, float, date. Фраза DEFAULT определяет значение по умолчанию. Это значение имеет более высокий приоритет, чем значение по умолчанию, указанное в домене (если вместо типа данных используется домен). Ограничения для таблицы (table_constraint) и ограничения для столбца (column_constraint), называемые также ограничениями целостности, накладывают определенные условия на вводимые в таблицу данные. Ограничения для столбца указываются непосредственно после описания столбца, а ограничения для таблицы - через запятую после описания любого столбца.

Ограничения для столбца могут указываться следующими фразами:

NOT NULL - в любой добавляемой или изменяемой строке столбец всегда должен иметь значение, отличное от NULL;
UNIQUE - все значения столбца должны быть уникальны; ^
PRIMARY KEY - устанавливает один столбец как первичный ключ и одновременно подразумевает, что все значения столбца будут уникальны;
CHECK (condition) - указываемое в скобках условие использует для сравнения значение столбца и возвращает TRUE, FALSE или UNKNOWN. Если при попытке выполнения SQL-оператора возвращаемое значение равно FALSE, то оператор выполнен не будет;
REFERENCES table (fields_list) - ограничение требует совпадения значений столбцов данной таблицы с указанными столбцами родительской таблицы.

Ограничения для таблицы могут указываться следующими фразами:

CHECK (condition) - указываемое в скобках условие использует для сравнения значение столбца и возвращает TRUE, FALSE или UNKNOWN. Если при попытке выполнения SQL-оператора возвращаемое значение равно FALSE, то оператор выполнен не будет;

^ **FOREIGN KEY** (fields_list) - это ограничение по внешнему ключу аналогично ограничению REFERENCES для столбцов и гарантирует, что все значения, указанные во внешнем ключе будут соответствовать значениям родительского ключа, обеспечивая ссылочную целостность. Следует отметить, что типы данных столбцов, используемых в этом ограничении, должны совпадать, а типы таблиц (постоянная базовая таблица, глобальная локальная временная таблица, локальная временная таблица) родительского и внешнего ключа - соответствовать друг другу.

Определение ограничений для таблицы

Объявление ограничений имеет в стандарте SQL92 следующее формальное описание:

table_constraint определяется как:

[CONSTRAINT constraint_name]
{ PRIMARY KEY (имя_поля .,…) }
| { UNIQUE (имя_поля .,…) } | { FOREIGN KEY (имя_поля .,…) }
{ REFERENCES имя_таблицы [(имя_поля .,…)] [*ref_specification*] }
| { CHECK (condition) }
[[NOT] DEFFERABLE]

column_constraint определяется как:

[CONSTRAINT constraint_name]
{ NOT NULL } | { PRIMARY KEY } | UNIQUE
| { REFERENCES имя_таблицы [(имя_поля .,…)]
[*ref_specification*]}
| { CHECK (condition) }
| [INITIALLY DEFFERED | INITIALLY IMMEDIATE]
[[NOT] DEFFERABLE]

ref_specification определяется как:

[MATCH {FULL | PARTIAL }]
[ON UPDATE
  { CASCADE | SET NULL | SET DEFAULT | NO ACTION }]
[ON DELETE
  { CASCADE | SET NULL | SET DEFAULT | NO ACTION }]

Ссылочная спецификация (ref_specification) определяет для ограничений FOREIGN KEY и REFERENCES тип совпадения и действия, выполняемые при внесении изменений в родительский ключ. Совпадение может быть определено как MATCH FULL (полное совпадение) или MATCH PARTIAL (частичное совпадение).
При полном совпадении значения родительского и внешнего ключей должны полностью совпадать, или все значения внешнего ключа должны быть NULL.
При частичном совпадении некоторые значения внешнего ключа могут быть равны NULL и значения каждой строки внешнего ключа, отличные от NULL, должны совпадать со значениями родительского ключа.
Если тип совпадения не указан, то предполагается что любое значение внешнего ключа присутствует в родительском ключе, но при этом во внешнем ключе допускаются значения NULL (частично или полностью).

Пример создания таблицы с помощью SQL-запроса:
Создать таблицу «Товары» со следующими атрибутами:

Атрибут	Тип атрибута	Формат атрибута
Серийный_номер	Текстовый	8
Модель	Текстовый	10

Таблица 1

Сделать атрибут товары первичным ключом.
SQL-запрос следующий:
CREATE TABLE Товары ([Серийный номер] CHAR(8) PRIMARY KEY, Модель CHAR(10) NOT NULL);

Примеры изменения атрибутов в таблице:
Добавить в таблицу «Товары» атрибут «Название». Тип и формат добавляемого атрибута такой же, как у атрибута «Модель».
SQL-запрос следующий:

```
ALTER TABLE Товары ADD COLUMN Название CHAR (10) NOT NULL;
```

Для удаления атрибута «Название» из таблицы «Товары» SQL-запрос следующий:

```
ALTER TABLE Товары DROP COLUMN Название;
```

Для изменения формата атрибута «Модель» с «10» на «20» SQL-запрос следующий:

```
ALTER TABLE Товары ALTER COLUMN Модель CHAR(20);
```

Для внесения, изменения и удаления данных в таблицах используется язык манипулирования данными (DML) с тремя основными командами – INSERT, UPDATE и DELETE соответственно.
Для добавления в таблицу «Товары» в атрибуты «Серийный_номер» и «модель» записи 11446804, PC-46C92HR используется следующий SQL-запрос:

```
INSERT INTO Товары ([Серийный_номер], [Модель]) VALUES ('11442804', 'PC-46C92HR');
```

Для изменения серийного номера 11442804 в модели PC-46C92HR на номер 22442804 таблицы «Товары» используется следующий SQL-запрос:

```
UPDATE Товары SET Товары.Серийный_номер = '22442804'
WHERE Товары.Модель='PC-46C92HR';
```

Для удаления записей модели PC-46C92HR из таблицы «Товары» используется следующий SQL-запрос:

```
DELETE Товары.Модель FROM Товары WHERE
Товары.Модель='PC-46C92HR';
```

СУБД Microsoft Access 2007 входит в состав пакета Microsoft Office. Запуск осуществляется через меню «Пуск» операционной системы Windows (Пуск/Все программы/Microsoft Office/ Microsoft Office Access 2007). После запуска СУБД появляется окно вида (рис.1):

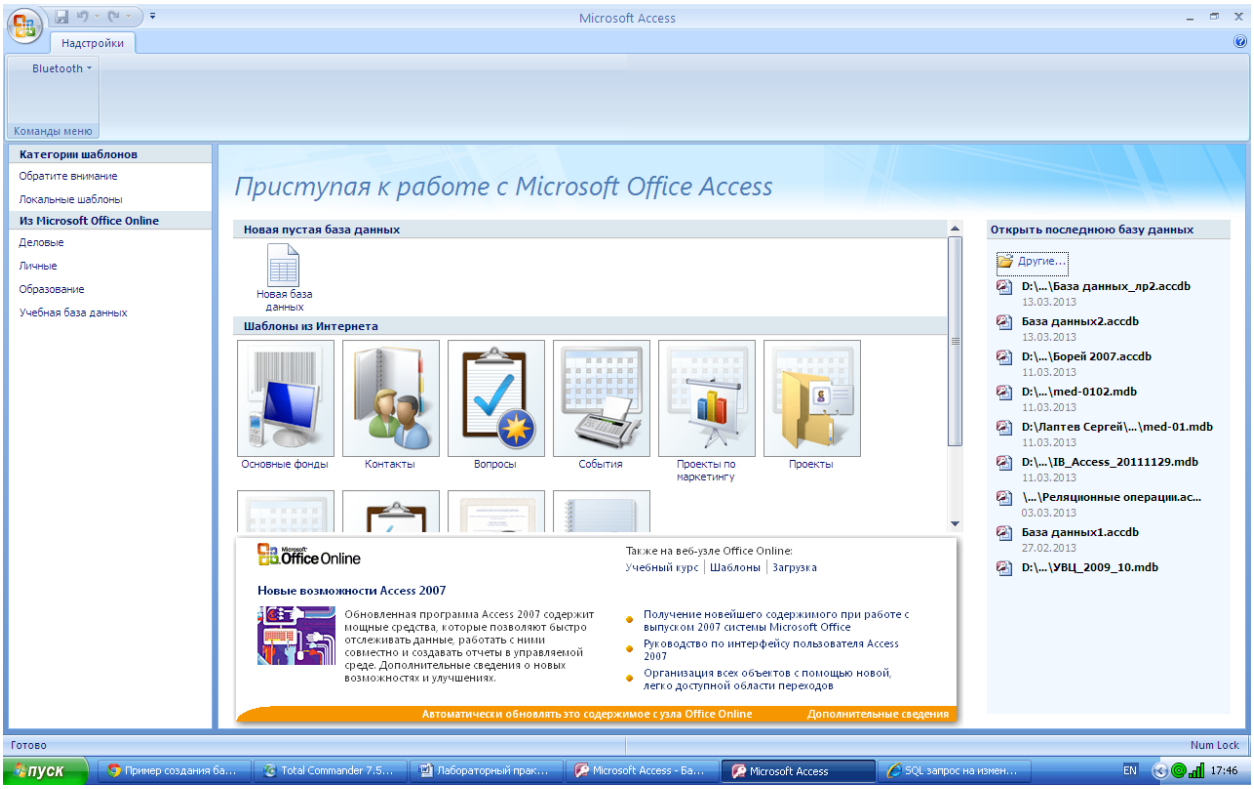

Рис. 1

Далее необходимо создать новую базу данных, щелкнув по значку «Новая база данных», ввести имя базы и путь к папке, где эта база будет сохранена (C:\WORK).

Для создания SQL-запросов в режиме SQL вначале нужно создать пустой запрос. Для этого в ленте нужно нажать вкладку «Создание», далее вкладку «Конструктор запросов». В итоге появится окно следующего вида (рис.2).

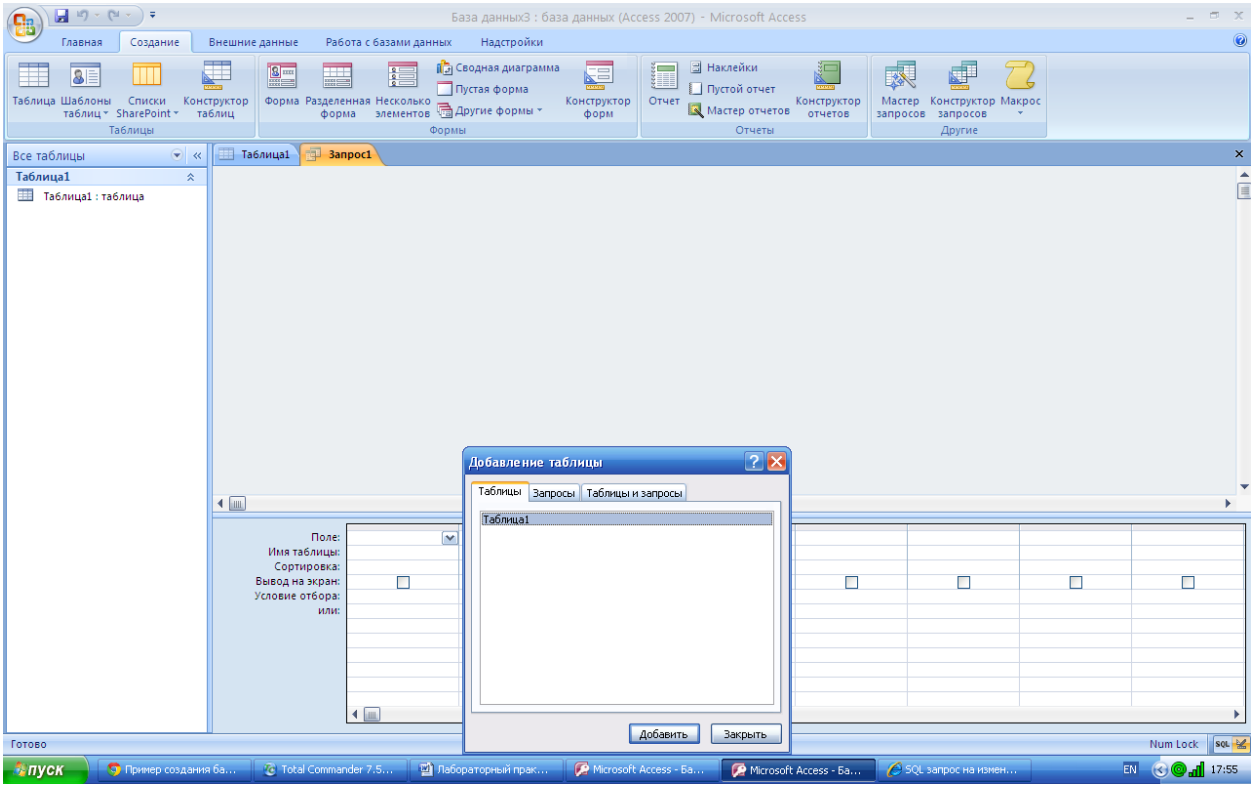

Рис.2

Окно «Добавление таблицы» необходимо закрыть, затем открыть вкладку SQL и выбрать «Режим SQL» (рис. 3)

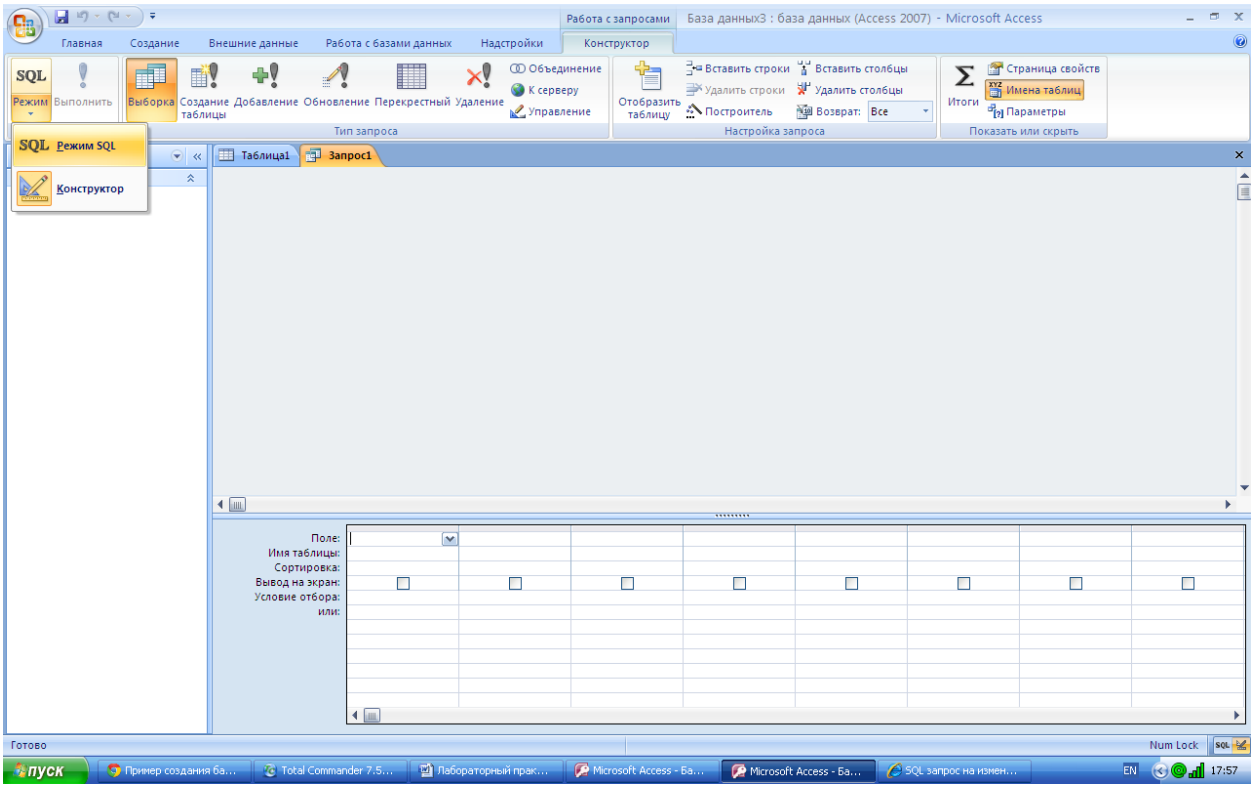

Рис. 3

Откроется окно ввода SQL-запросов.

После ввода текста запроса для его выполнения необходимо нажать вкладку «Выполнить» (рис.4).

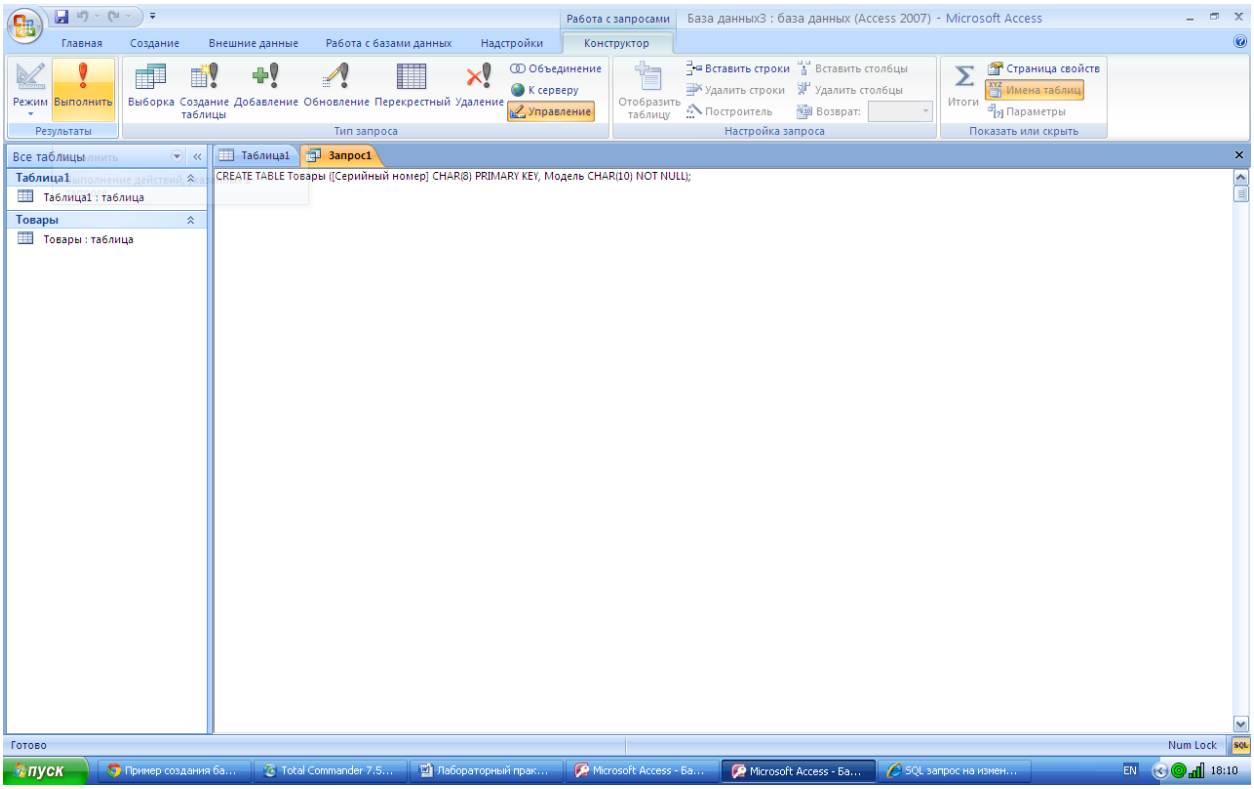

Рис. 4

Для сохранения запроса при закрытии окна запроса необходимо в ответ на запрос системы о сохранении ввести имя запроса, после чего он будет сохранен и появится в левом окне системы.
Для выполнения уже имеющегося запроса необходимо открыть его в режиме конструктора, щелкнув по нему правой кнопкой мыши и выбрав «Конструктор» в выпадающем меню. Далее в ленте выбрать «Конструктор» и щелкнуть по пиктограмме «Выполнить».

Задание

1. Создать таблицу «Товары» с атрибутами и форматами атрибутов согласно Таблицы 1. Сохраните запрос на создание таблицы с именем «Запрос на создание таблицы».
2. Выполните все запросы изменения атрибутов, указанные в примерах SQL-запросов на изменения атрибутов в таблице. Сохраните запросы соответственно с именами «Добавление атрибутов», «Удаление атрибутов», «Изменение атрибутов». Покажите результат преподавателю.

3. Выполните все запросы внесения, изменения и удаления данных в таблице, указанные в примерах SQL-запросов манипулирования данных в таблице. Сохраните запросы соответственно с именами «Добавление данных», «Изменение данных», «Удаление данных». Покажите результат преподавателю.

Лабораторная работа №3.
(SQL) запросы связей между таблицами в многотабличных базах данных под управлением СУБД Access.

При проектировании базы данных, используются различные таблицы для разных данных. Пример: клиенты, заказы, записи, сообщения и т.д. Так же нужны взаимосвязи между этими таблицами. Например, клиент имеет заказы, а у заказа есть позиции (товары). Эти взаимосвязи должны быть отражены в базе данных. А также, когда мы получаем данные с помощью SQL, мы должны использовать определенные типы запросов JOIN, чтобы получить нужный результат.

На практике наиболее часто используются следующие типы связей:

- связь "один к одному" , 1:1;
- связь "один ко многим", 1:N;
- связь "многие ко многим", N:M.

Когда данные выбираются из нескольких связанных таблиц, используется запрос *JOIN*. Есть несколько типов присоединения. В данном курсе будут рассматриваться следующие:

- Cross Joins (Перекрестное соединение)
- Natural Joins (Естественное соединений)
- Inner Joins (Внутреннее соединений)
- Left (Outer) Joins (Левое (внешнее) соединение)
- Right (Outer) Joins (Правое (внешнее) соединение)

Рассмотрим вначале примеры связей.

Связь один к одному

Пусть есть таблица покупателей
Покупатели

Покупатель_id	Покупатель_Имя	Покупатель_адрес
101	Иванов Александр	г. Краснодар, Мира 10
102	Петров Павел	г. Москва, Горького 15

Информацию об адресе покупателя можно расположить в другой таблице, введя идентификатор адреса в первую:

Покупатель_id	Покупатель_Имя	Адрес_id
101	Иванов Александр	301
102	Петров Павел	302

Адрес_id	Покупатель_адрес
301	г. Краснодар, Мира 10
302	г. Москва, Горького 15

Теперь у нас есть связь между таблицами покупателей и адресами. Если каждый адрес может принадлежать только одному покупателю, то такая связь называется "Один к одному". Такой тип отношений не очень распространен. Первоначальной таблицы, в которой информация о покупателе и его адресе хранилась вместе, в большинстве случаев достаточно.
Схематично связь один к одному можно представить следующим рисунком:

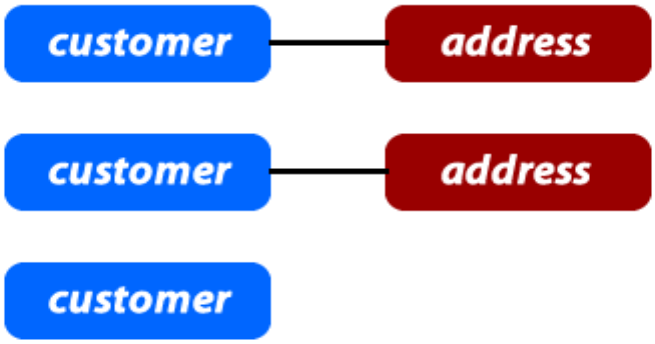

Рис. 1
Customer – покупатель, address – адрес.

Обратите внимание, что существование данных отношений не обязательно, например, может существовать запись о покупателе без связанной записи о его адресе.

Связь один ко многим и многие к одному

Этот тип отношений наиболее часто встречающийся. Рассмотрим пример:

- У покупателей может быть несколько заказов.
- Заказ может содержать несколько товаров.
- Товары могут иметь описание на нескольких языках.

В этих случаях нам потребуется создать связь "Один ко многим".

Пример:
Покупатели

Покупатель_id	Покупатель_Имя
101	Иванов Александр
102	Петров Павел

Заказы

Заказ_id	Покупатель_id	Заказ_дата	Цена
555	101	24/12/12	4000
556	102	25/12/12	5600
557	101	26/12/12	3300

Каждый покупатель может иметь 0 или более заказов. Но каждый заказ может принадлежать только одному покупателю.
Схематично связь один ко многим можно представить следующим рисунком:

Схематично связь один к одному можно представить следующим рисунком:

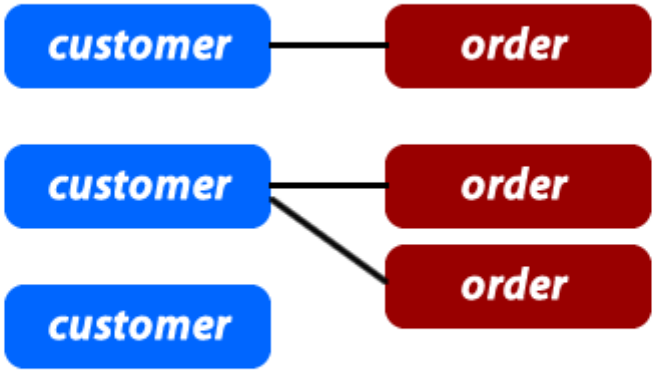

Рис. 2
Customer – покупатель, order – заказ.

Связь многие ко многим

В некоторых случаях требуется многочисленные связи по обе стороны отношений. Например, каждый заказ может содержать множество товаров. И каждый товар может присутствовать во многих заказах.

Для такой связи потребуется создать дополнительную таблицу:

Заказы

Заказ_id	Покупатель_id	Заказ_дата	Цена
555	101	24/12/12	4000
556	102	25/12/12	5600

Пункты

Пункт_id	Пункт_имя	Пукт_описание
201	Телефон	Две сим-карты
202	Монитор	Жидкокристаллический
203	Ноутбук	Процессор Intel

Пункты_заказы

Заказ_id	Пнкт_id
555	201
555	202
556	202
556	203

Назначение таблицы «Пункты_Заказы» только одно - создать связь "Многие ко многим" между товарами и заказами.

Схематично связь многие ко многим можно представить следующим рисунком:

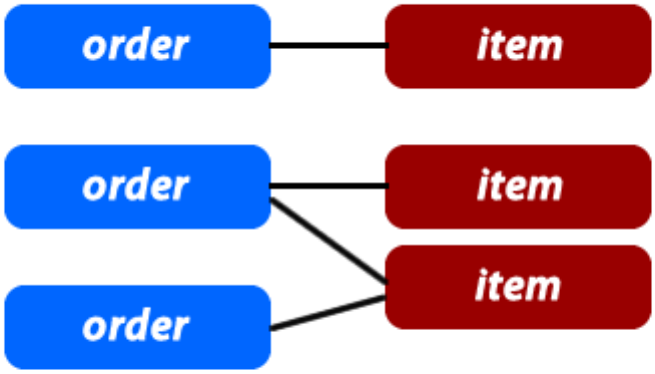

Рис. 3
Item – пункт.
Если к рисунку добавить записи Пункт_заказы, то он будет выглядеть так:

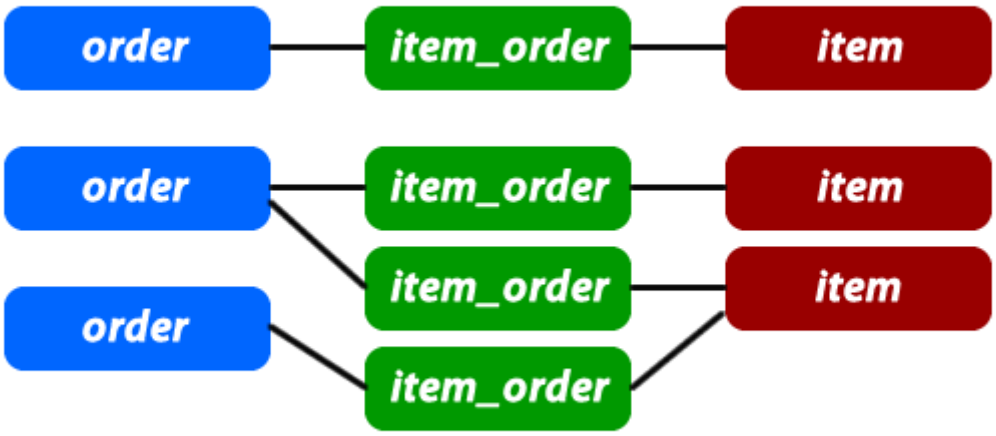

Рис. 4

Внешние ключи.

В таблицах, рассмотренных выше, всегда было поле вида «****_id», которое ссылалось на столбец в другой таблице. В примере связей один ко многим столбец Покупатель_id, в таблице Заказы, является внешним ключом:

Покупатели

Покупатель_id	Покупатель_Имя
101	Иванов Александр
102	Петров Павел

Заказы

Заказ_id	Покупатель_id	Заказ_дата	Цена
555	101	24/12/12	4000
556	102	25/12/12	5600
557	101	26/12/12	3300

Создадим SQL-запрос, который задает внешний ключ для таблицы Заказы и устанавливает связь один ко многим между таблицами Покупатели и Заказы.

Вначале создадим простую таблицу Покупатели.

SQL-запрос следующий:

CREATE TABLE Покупатели (Покупатель_id INT PRIMARY KEY, Покупатель_имя VARCHAR(100));

Теперь создадим таблицу Товары, которая будет содержать вторичный ключ.

SQL-запрос следующий:

CREATE TABLE Заказы (Заказ_id INT PRIMARY KEY, Покупатель_id INT, Цена DOUBLE, FOREIGN KEY (Покупатель_id) REFERENCES Покупатели(Покупатель_id));

Оба столбца (Покупатели.Покупатель_id и Заказы. Покупатель_id) должны быть одного типа. Например, если у первого тип INT, то и у второго должен быть тип INT.

После выполнения этих двух в Схеме данных СУБД Access данные таблицы появятся со связью один ко многим (рис. 5). Для просмотра схемы данных необходимо в ленте щелкнуть по вкладке «Работа с базами данных» и выбрать пиктограмму «Схема данных».

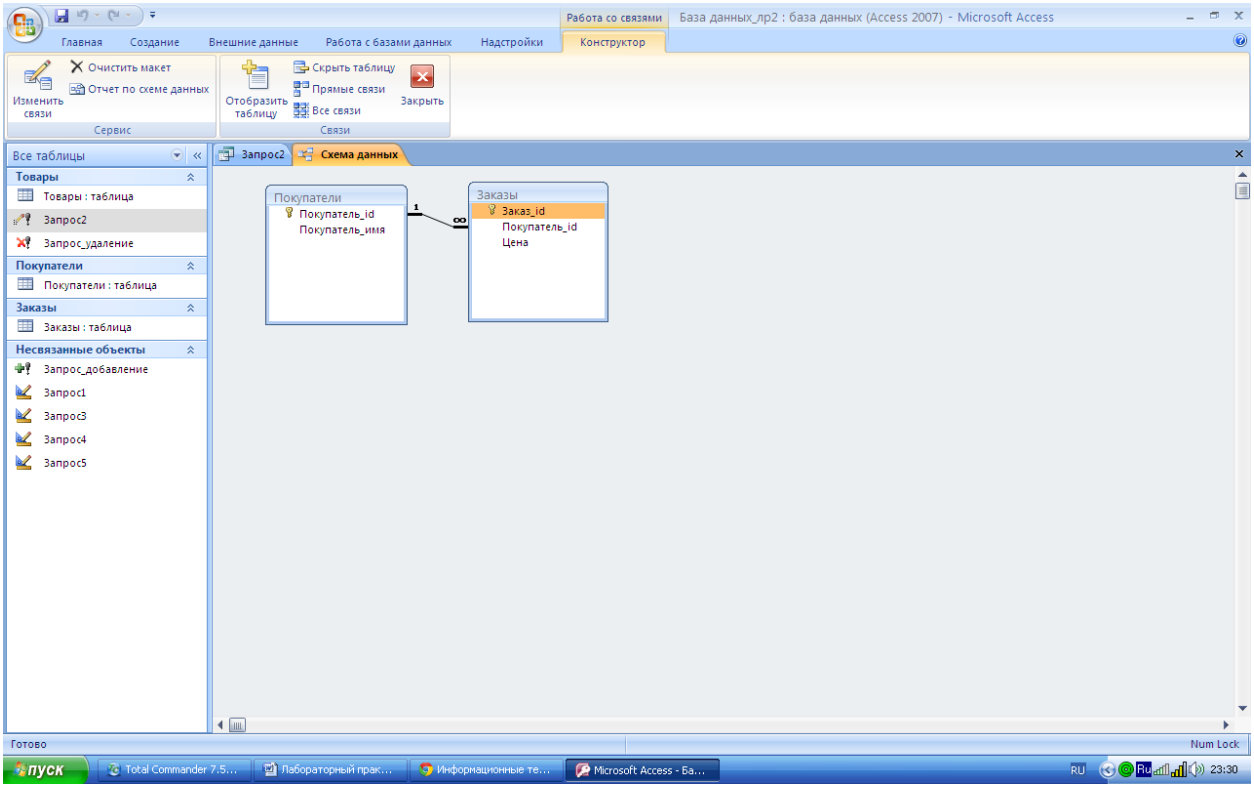

Рис. 5

В рассмотренном примере связь между таблицами была создана в запросе создания таблицы Заказы, то есть внутри запроса на создание таблицы был также подзапрос на создание связи между двумя таблицами. Созданная связь является объектом базы данных, у этого объекта есть имя, которое СУБД присваивает ему автоматически. На практике может возникнуть ситуация, когда созданную связь необходимо удалить (например, в случае, если случайно выбрано не то поле для связи и выполнен запрос на создание). Для того, чтобы удалить связь, необходимо вначале выяснить имя данного объекта СУБД. Для того, чтобы узнать имя связи, необходимо выполнить следующий SQL-запрос:

```
SELECT * FROM MSysRelationships;
```

После выполнения данного запроса в рассматриваемом примере появится таблица (рис.6):

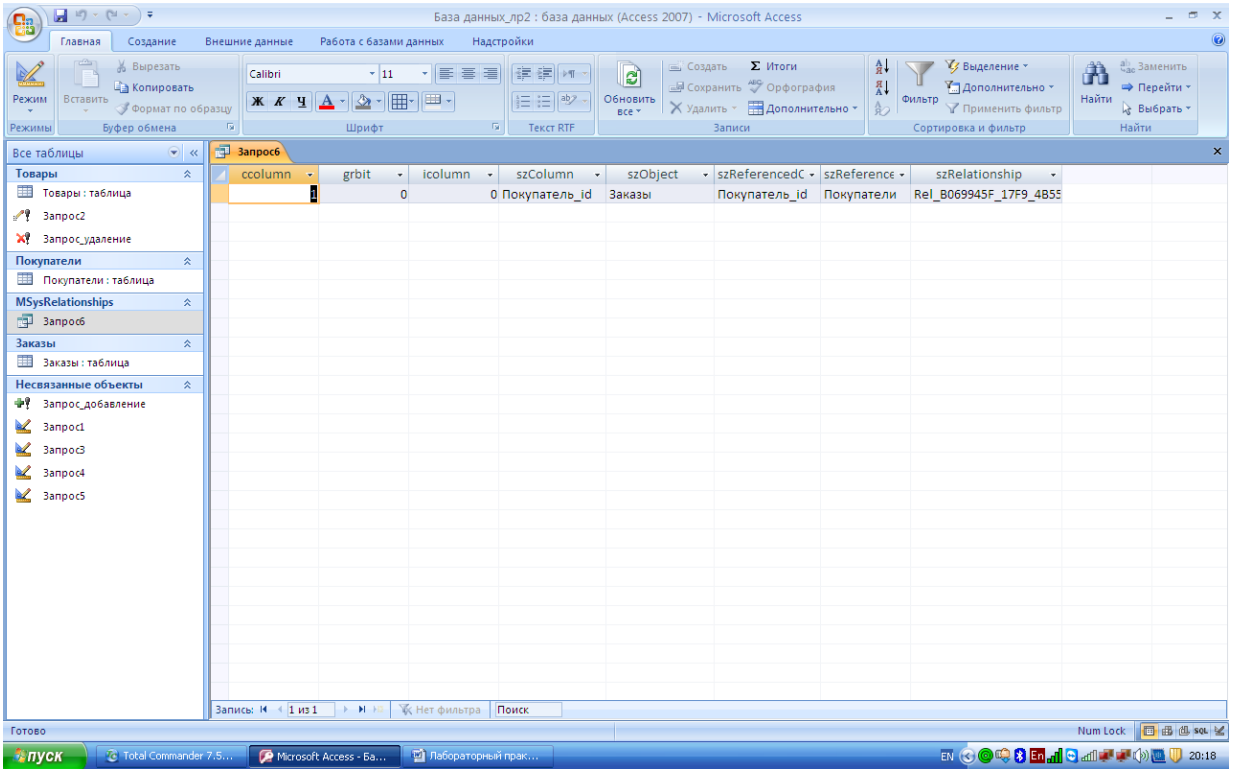

Рис. 6

В поле szRelationship содержится искомое имя созданной связи. В данном случае имя связи - Rel_B069945F_17F9_4B55. Теперь, для того чтобы удалить связь с известным именем, необходимо выполнить следующий SQL-запрос:

```
ALTER TABLE Заказы DROP CONSTRAINT
Rel_B069945F_17F9_4B55;
```

Если теперь вновь открыть схему данных, то на ней связи между таблицами уже не будет, так как она была удалена (рис. 7).

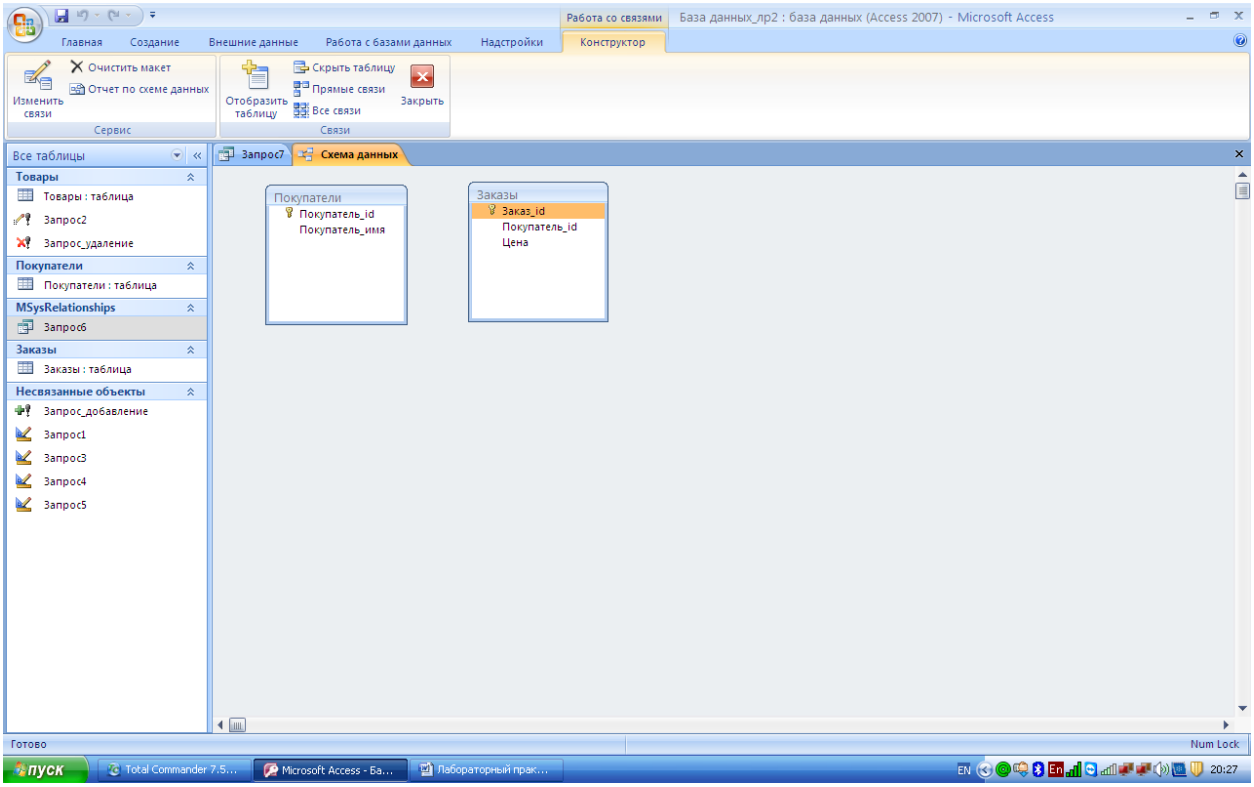

Рис. 7

В ситуации, когда таблицы уже созданы и есть ключевые поля, можно создать связь между ними и присвоить ей имя вручную. В рассматриваемом примере связь один ко многим между таблицами Покупатели и Заказы по полю Покупатель_id с именем Relation1 создается следующим SQL-запросом:

ALTER TABLE Заказы ADD CONSTRAINT Relation1 FOREIGN KEY ([Покупатель_id]) REFERENCES Покупатели ([Покупатель_id]);

Для удаления связи с именем Relation1 между вышеуказанными таблицами используется следующий SQL-запрос

ALTER TABLE Заказы DROP CONSTRAINT Relation1;

В лабораторной работе № 2 были рассмотрены запросы на создание ключевого поля (первичный ключ) при создании таблицы. Рассмотрим теперь примеры SQL-запросов, которые позволяют задать ключевое поле в уже созданной таблице и удалить ключ.
Когда в СУБД задается ключевое поле, автоматически создается индекс (таблица, в которой в упорядоченном по значениям полей

виде хранятся ссылки на записи). Для удаления индекса необходимо вначале узнать его имя, а затем выполнить SQL-запрос на удаление индекса с указанным именем. Удаление происходит с использованием удаления ограничения (CONSTRAINT), которое индекс накладывает на ключевое поле. Для того, чтобы узнать имя индекса, необходимо открыть таблицу в режиме конструктора, далее щелкнуть мышью по пиктограмме Индексы в ленте, после чего откроется окно с названиями и описаниями индексов (рис. 8).

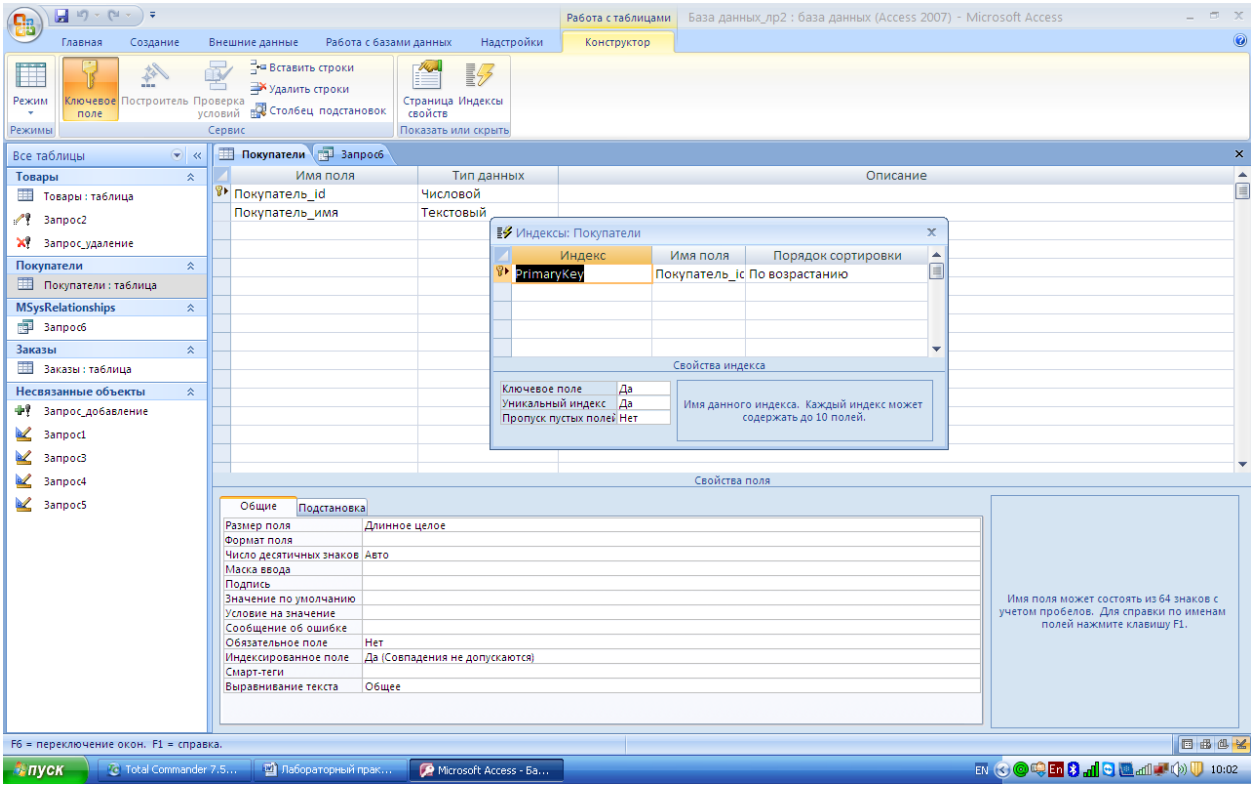

Рис. 8

Для удаления индекса с именем PrimaryKey из таблицы Покупатели используется следующий SQL-запрос

ALTER TABLE Покупатели DROP CONSTRAINT PrimaryKey;

Задание

1. Спроектировать базу данных «Сведения о клиентах и заказах», содержащую следующие таблицы:

а) Клиенты. Поля таблицы: Код клиента, Название, Адрес счета, Город, Регион, Индекс, Страна, Телефон, Факс, Сумма долга;

б) Сотрудники. Поля таблицы: Код сотрудника, Имя, Фамилия, Должность;

в) Заказы. Поля таблицы: Номер заказа, Дата заказа, Код сотрудника, Код клиента, Серийный номер, Дата завершения, Ставка налога;

г) Товары. Поля таблицы: Серийный номер, Модель;

д) Модели. Поля таблицы: Модель, Описание, Год разработки.

Поля и типы значений полей:

Атрибут	**Тип атрибута**	**Формат атрибута**
Код клиента	Числовой	Длинное целое
Название	Текстовый	30
Имя	Текстовый	15
Фамилия	Текстовый	15
Адрес счета	Текстовый	20
Город	Текстовый	15
Регион	Текстовый	15
Индекс	Текстовый	6
Страна	Текстовый	15
Должность	Текстовый	20
Телефон	Текстовый	10
Факс	Текстовый	10
Код заказа	Числовой	Длинное целое
Код сотрудника	Числовой	Длинное целое
Номер заказа	Числовой	Длинное целое
Дата заказа	Дата	Дата/время
Модель	Текстовый	10
Серийный номер	Текстовый	8
Описание	Текстовый	50
Дата завершения	Дата	Дата/время
Ставка налога	Числовой	Длинное целое

Ключи и связи между таблицами должны быть определены следующим образом:

В таблице Клиенты первичный ключ - Код клиента.
Во таблице Сотрудники первичный ключ- Код сотрудника.
В таблице Заказы первичный ключ- Номер заказа. Внешние ключи - Код сотрудника, Код клиента, Серийный номер.
В таблице Товары первичный ключ- Серийный номер, внешний ключ - Модель.
В таблице Модели первичный ключ- Модель.

Если правильно созданы таблицы и связи, то в результате должна получиться схема данных следующего вида (рис. 9):

Рис. 9

2. Заполнить таблицы базы данных записями согласно рисункам, представленным ниже

Клиенты

Код клиент	Название	Адрес счета	Город	Телефон	Индекс	Страна
1	ТелеСтиль	70452012222000008412	Тверь	441289	172530	Россия
2	Плазмацентр	71451201401402211551	Тверь	225641	172530	Россия
3	Телерынок	79210155000140114455	Королев	442150	141080	Россия
4	Фостергруп	79820401440014012225	Москва	407525	121500	Россия
5	Мвидео	71240455045023601404	Москва	225021	121500	Россия
6	Эльдорадо	73601044204110004004	Москва	336512	121500	Россия
7	Техносила	70941001400040457045	Королев	225140	141080	Россия
8	Наномир	70365001000040045891	Королев	265412	141080	Россия
9	ТехноЦентр	70014580120445842321	Тверь	331716	172530	Россия
10	Цифроград	73601450159877523012	Орел	445714	437000	Россия
0						

Рис. 10

Сотрудники

Код сотрудника	Имя	Фамилия	Должность	Щелкните для добавления
1	Александр	Плюснин	продавец 1кат.	
2	Николай	Балышев	продавец 1кат.	
3	Мария	Шатова	продавец 1кат.	
4	Андрей	Ветров	продавец 2кат.	
5	Егор	Метель	продавец 2кат.	
6	Вячеслав	Караваев	продавец 1кат.	
8	Елена	Архипова	продавец 2кат.	
9	Артем	Хорошев	продавец 2кат.	
10	Дмитрий	Зубков	продавец 2кат.	
0				

Рис. 11

Заказы

Номер зака	Дата заказа	Код сотруд	Код клиент	Серийный н	Дата завершен	Ставка налс	Щелкните для добавления
2	15.02.2009	6	8	18012205	16.02.2009	20	
3	16.02.2009	8	3	36512001	16.02.2009	20	
5	16.02.2009	10	1	21777745	19.02.2009	20	
6	19.02.2009	2	2	11446804	20.02.2009	16	
7	19.02.2009	5	2	13241114	19.02.2009	20	
8	19.02.2009	6	4	40110057	24.02.2009	16	
9	22.02.2009	9	5	36985014	24.02.2009	20	
11	22.02.2009	8	9	31548722	22.02.2009	20	
0		0	0			0	

Рис. 12

Товары

Серийный ном	Модель	Щелкните для добавления
11446804	PC-46C92HR	
13241114	UE46B600W	
18012205	LE37B530W	
21777745	PC-50C92HR	
31420145	LE46B750U	
31548722	LE32B750U	
35921454	UE46B600W	
36512001	PC-46C92HR	
36910141	LE26B460W	
36985014	UE26B600W	
40110057	UE32B600W	
44144255	UE46B600W	
65677156	PC-63P76FD	

Рис. 13

Модели

Модель	Описание	Выпуск	Щелкните для добавления
LE26B460W	ЖК 26" 1920*1080 выс. Контр	2010	
LE32B750U	ЖК 32" 1920*1080	2011	
LE37B530W	ЖК 37" 1920*1080 выс. Контр	2011	
LE46B750U	ЖК 46" 1920*1080	2011	
PC-46C92HR	плазменный 46" 1365*768	2011	
PC-50C92HR	плазменный 50" 1365*768	2012	
PC-63P76FD	плазменный 63" 1920*1080	2012	
UE26B600W	ультратонкий ЖК 26" 1920*1080 М	2010	
UE32B600W	ультратонкий ЖК 32" 1920*1080 М	2010	
UE46B600W	ультратонкий ЖК 46" 1920*1080 М	2011	

Рис. 14

По окончанию выполнения заданий должны быть получены следующие результаты:

Первая часть - SQL-запросы на создание таблиц и связей с соответствующими названиями (минимум 5 запросов на создание таблиц, если ключи и связи созданы сразу). Если вначале созданы таблицы, а затем ключи им связи, то помимо 5 запросов на создание таблиц должны быть запросы на создание ключей и связей с соответствующими названиями.

Вторая часть – заполненные данными таблицы в соответствии с рисунками и 5 SQL-запросов на вставку данных в соответствующие таблицы с соответствующими названиями.

База данных обязательно должна быть сохранена на электронный носитель информации.

Лабораторная работа №4.
(SQL) запросы реляционных операций в базах данных под управлением СУБД Access.

Виды действий (манипуляций) над данными в реляционной модели представляют собой множество операций, получивших в совокупности название *реляционной алгебры*, и реляционной операции присваивания. Операция присваивания производит присваивание значения некоторого производного выражения реляционной алгебры другому отношению.

Каждая операция реляционной алгебры использует одну или две таблицы в качестве операндов и создает в результате некоторую новую таблицу. В реляционной алгебре были определены восемь таких операций, объединенные в две группы по четыре операции в каждой.

Первая группа - *традиционные теоретико-множественные операции*.

В каждой из этих операций используется два операнда (таблицы).

Для всех операций, кроме декартова произведения, эти два операнда должны быть совместимы по объединению, т.е. они должны быть одной степени и их *i*-е атрибуты ($i = \overline{1,n}$) должны быть связаны с одним и тем же доменом.

Рассмотрим примеры традиционных операций реляционной базы данных.

Операция объединение. Объединением двух отношений A и B называется множество всех кортежей *t*, принадлежащих либо A, либо B, либо им обоим.

Результатом объединения отношений A и B будет отношение с тем же заголовком, что и у совместимых по типу отношений A и B, и телом, состоящим из кортежей, принадлежащих или A, или B, или обоим отношениям. Пусть даны следующие отношения:

Персоны

Имя	Возраст	Вес
Александр	34	80
Иван	28	64
Георгий	29	70
Елена	54	54
Петр	34	80

Персонажи

Имя	Возраст	Вес
Дональд	24	19
Том	25	23
Джерри	81	27

Результатом объединения этих отношений будет отношение:

Имя	Возраст	Вес
Александр	34	80
Георгий	29	70
Джерри	81	27
Дональд	24	19
Елена	54	54
Иван	28	64
Петр	34	80
Том	25	23

SQL-запрос для объединения данных отношений следующий:

```
SELECT Имя, Возраст, Вес FROM Персоны
UNION
SELECT Имя, Возраст, Вес FROM Персонажи;
```

Операция пересечение Пересечением двух отношений A и B называется множество всех кортежей *t*, каждый из которых принадлежит как A, так и B.

Результатом пересечения отношений A и B будет отношение с тем же заголовком, что и у отношений A и B, и телом, состоящим из кортежей, принадлежащих одновременно обоим отношениям A и B. Пусть даны следующие отношения:

Персоны

Имя	Возраст	Вес
Александр	34	80
Иван	28	64
Георгий	29	70
Елена	54	54
Петр	34	80

Персонажи

Имя	Возраст	Вес
Дональд	24	19
Том	25	23
Джерри	81	27
Георгий	29	70
Елена	54	54

Результатом пересечения этих отношений будет отношение:

Имя	Возраст	Вес
Георгий	29	70
Елена	54	54

SQL-запрос для пересечения данных отношений следующий:

```
SELECT Имя, Возраст, Вес FROM Персоны
INTERSECT
SELECT Имя, Возраст, Вес FROM Персонажи;
```

Следует отметить, что в СУБД Access данный SQL-запрос не поддерживается, что является ее недостатком. Также в СУБД Access не поддерживаются SQL-запросы для разности отношений.

Операция разность. Разностью между двумя отношениями A и B называется множество всех кортежей *t*, каждый из которых принадлежит A и не принадлежит B.

Результатом разности отношений A и B будет отношение с тем же заголовком, что и у совместимых по типу отношений A и B,

и телом, состоящим из кортежей, принадлежащих отношению A и не принадлежащих отношению B.

Результатом разности отношений Персон и Персонажей предыдущего примера будет отношение вида:

Имя	Возраст	Вес
Александр	34	80
Иван	28	64
Петр	34	80

SQL-запрос для разности отношений следующий:

```
SELECT Имя, Возраст, Вес FROM Персоны
EXCEPT
SELECT Имя, Возраст, Вес FROM Персонажи;
```

Операция *декартово произведение*. Декартовым произведением двух отношений A и B называется множество всех кортежей *t*, таких, что *t* является конкатенацией некоторого кортежа *a*, принадлежащего A, и какого-либо кортежа *b*, принадлежащего B. (конкатенация - это соединение в цепочки).

При выполнении декартового произведения двух отношений получается отношение, кортежи которого являются конкатенацией (сцеплением) кортежей первого и второго операндов. Пусть даны следующие отношения:

Мультфильмы

Код мультфильма	Название мультфильма
0	Симпсоны
1	Том и Джерри
2	Ну погоди

Каналы

Код канала	Название канала
0	СТС
1	2x2

Результатом декартового произведения отношений Мультфильмы и Каналы будет отношение вида:

Код мультфильма	Название мультфильма	Код канала	Название канала
0	Симпсоны	0	СТС
0	Симпсоны	1	2х2
1	Том и Джерри	0	СТС
1	Том и Джерри	1	2х2
2	Ну погоди	0	СТС
2	Ну погоди	1	2х2

SQL-запрос для декартового произведения вышеуказанных отношений следующий:

SELECT * FROM Мультфильмы, Каналы;

Здесь «*» означает, что выбираются все поля из первой и второй таблиц.

Вторая группа реляционных операций – специальные реляционные операции.

Операция *селекция*. Пусть *theta* представляет собой любой достижимый оператор сравнения данных, например, $=, \neq, >, \geq, \leq$ и т.д.

Предика́т (*n*-местный, или n-арный) — это функция с множеством значений {0,1} (или «ложь», и «истина»), определённая на множестве $M = M_1 \times M_2 \times ... \times M_n$. Таким образом, каждый набор элементов множества *M* характеризуется либо как «истинный», либо как «ложный».

Theta - селекцией отношения A по атрибутам *x* и *y* называется множество всех кортежей *t* из A, таких, что истинен предикат «*t.x theta t.y*». Атрибуты *x* и *y* должны быть определены на одном и том же домене, и для этого домена оператор *theta* должен иметь смысл. Вместо атрибута *y* может быть задана константа (например, выбрать из платежной ведомости записи о своих сотрудниках

имеющих зарплату 500 руб.). Таким образом, оператор *theta* - селекции позволит получать «горизонтальные» подмножества заданного отношения, т.е. подмножество таких кортежей заданного отношения, для которых выполняются поставленное условие.

Результатом селекции (выборки) из некоторого отношения R будет отношение, для которого выполняется заданное данной выборкой условие. Пусть задано отношение:

Персоны

Имя	Возраст	Вес
Александр	34	80
Иван	28	64
Георгий	29	70
Елена	54	54
Петр	34	80

Необходимо выбрать те записи отношения, у которых возраст не моложе 34 лет (Возраст>=34).

Результатом селекции будет следующее отношение:

Имя	Возраст	Вес
Александр	34	80
Елена	54	54
Петр	34	80

SQL-запрос для данной выборки следующий:

```
SELECT * FROM Персоны WHERE Возраст>=34;
```

Операция *проекция*. Операция проекция позволяет получить «вертикальное» подмножество заданного отношения, т.е. такое подмножество, которое получается выбором специфицированных (определенных) атрибутов с последующим исключением, если это необходимо, избыточных дубликатов кортежей, состоящих из значений выбранных атрибутов.

Результатом проекции из некоторого отношения R является выбор определенных полей с удалением, если это необходимо,

избыточных дубликатов записей, состоящих из одинаковых значений выбранных полей.

Пусть нужно выбрать из вышеуказанного отношения Персоны поля Возраст, Вес. Тогда результатом данной проекции будет следующее отношение:

Возраст	Вес
28	64
29	70
34	80
54	54

SQL-запрос для данной проекции следующий:

```
SELECT DISTINCT Возраст, Вес FROM Персоны;
```

Примечательно, что в SQL для полного соответствия операции проекции необходимо указывать ключевое слово `DISTINCT`, поскольку без него строка с возрастом 34 и весом 80 отобразится дважды.

Операция *соединение*. Пусть *theta* имеет тот же смысл, что и в операции селекции. Тогда *theta* - соединением отношения A по атрибуту *x* с отношением B по атрибуту *y* называется множество всех кортежей *t*, таких, что *t* является конкатенацией какого-либо кортежа *a*, принадлежащего A, и какого-либо кортежа *b*, принадлежащего B, и предикат «*a.x theta b.y*» принимает значение „истина“. При этом атрибуты A.*x* и B.*y* должны быть определены на одном и том же домене, а оператор *theta* должен иметь смысл для этого домена. Если оператор *theta* - равенство, то соединение называется *экви-соединением*.. Из этого определения следует, что результат экви-соединения должен включать два идентичных атрибута. Если один из этих атрибутов исключается, что можно осуществить с помощью проекции, результат называется *естественным соединением*. Под неуточненным термином «соединение» понимают естественное соединение. Операция «соединение» похожа на декартово произведение. Отличие состоит в том, что декартово произведение предполагает сцепление

каждого кортежа из отношения A с каждым кортежем из B, а в операции соединения кортеж из отношения A сцепляется только с теми кортежами из B, для которых выполнено условие, например, «a.*x* = *b.y*».

Операция соединения есть результат последовательного применения операций декартового произведения и выборки. Если в отношениях и имеются атрибуты с одинаковыми наименованиями, то перед выполнением соединения такие атрибуты необходимо переименовать. Пусть заданы два отношения R_1 и R_2 вида:

R_1 (мультфильмы)

Код мультфильма	Название мультфильма	Название канала
0	Симпсоны	2x2
1	Том и Джери	2x2
2	Ну погоди	RenTV

R_2 (каналы)

Код канала	Частота
RenTV	3,14
2x2	783,25

Выполняется соединение этих двух отношений с условием совпадения названия канала с кодом канала (код канала = название канала).

Первый этап – декартово произведение. Результат произведения – отношение R_3.

Код мультфильма	Название мультфильма	Название канала	Код канала	Частота
0	Симпсоны	2x2	RenTV	3,14
0	Симпсоны	2x2	2x2	783,25
1	Том и Джери	2x2	RenTV	3,14
1	Том и Джери	2x2	2x2	783,25
2	Ну погоди	RenTV	RenTV	3,14
2	Ну погоди	RenTV	2x2	783,25

Второй этап – выборка. В итоге результатом соединения вышеуказанных отношений будет отношение:

Код мультфильма	Название мультфильма	Название канала	Код канала	Частота
0	Симпсоны	2x2	2x2	783,25
1	Том и Джери	2x2	2x2	783,25
2	Ну погоди	RenTV	RenTV	3,14

SQL-запрос для данной операции соединения следующий:

```
SELECT *
FROM Мультфильмы, Каналы
WHERE Название_канала=Код_канала;
```

Операция *деление*. В простейшей форме операция деления делит отношение степени два (делимое) на отношение степени один (делитель) и создает (продуцирует) результирующее отношение степени один (частное). Пусть делимое A имеет атрибуты *x* и *y*, а делитель B атрибут *y* (рисунок выше г). Атрибуты A.*y* и B.*y* должны быть определены на одном домене. Результатом деления A на B является отношение C с единственным атрибутом *x*, таким, что каждое значение *x* этого атрибута C.*x* появляется как значение A.*x*, а пара значений (*x*,*y*) входит в A для *всех* значений *y*, входящих в B. Другими словами, кортеж включается в результирующее отношение C только в том случае, если его декартово произведение с отношением B содержит отношение A.

Реляционное деление достаточно нетривиально описать, но на примере его смысл нагляден. В целом, из таблицы A берутся значения строк, для которых присутствуют все комбинации значений из таблицы B. Пример деления:

Пусть заданы два соотношения R_1 и R_2 вида:

R_1 (мультфильмы)

Код мультфильма	Название мультфильма	Название канала
0	Симпсоны	RenTV
0	Симпсоны	2x2
0	Симпсоны	СТС
1	Том и Джери	RenTV
1	Том и Джери	2x2
2	Ну погоди	СТС
2	Ну погоди	2x2

R_2 (каналы)

Название канала
RenTV
2x2

При делении R_1 на R_2 результатом будет

Код мультфильма	Название мультфильма
0	Симпсоны
1	Том и Джери

«Симпсоны» и «Том и Джери» - мультфильмы, которые показывались на RenTV и 2x2. «Ну погоди» не показывался по RenTV, поэтому был исключен из результирующей таблицы.
Операция деления на практике используется достаточно редко, простейшего SQL-запроса операции деления не существует.

Задание

1. Создайте таблицы «Персоны» и «Персонажи», представленные в описании операции объединения. Создание и заполнение таблиц можно выполнить в режиме конструктора. Выполните SQL-запрос на создание объединения данных таблиц. Проверьте совпадение

полученного результата объединения с таблицей, полученной в описании объединения таблиц. Сохраните запрос на объединение таблиц с именем «Запрос на объединение».

2. Создайте таблицы «Мультфильмы» и «Каналы», представленные в описании операции декартового произведения. Выполните SQL-запрос на создание декартового произведения таблиц и проверьте совпадение полученного результата с результатом, полученным в описании декартового произведения. Сохраните запрос на создание декартового произведения таблиц с именем «Запрос декартово произведение».
3. Выполните запрос на выборку из таблицы Персоны. Сравните результат выполнения запроса с представленным в описании. Сохраните запрос на выборку с именем «Запрос на выборку».
4. Выполните SQL-запрос на создание проекции из таблицы Персоны. Сравните результат выполнения запроса с представленным в описании. Сохраните запрос на создание проекции с именем «Запрос-проекция».
5. Создайте таблицы «Мультфильмы_R1» и «Каналы_R1», представленные в описании операции соединения. Выполните SQL-запрос на создание соединения. Проверьте совпадение полученного результата соединения с таблицей, полученной в описании соединения таблиц. Сохраните запрос с именем «Запрос-соединение».

Результаты выполнения работы покажите преподавателю.

Лабораторная работа №5.
(SQL) запросы выборки данных в СУБД Access.

Для выборки данных из БД используется язык запросов (DQL), наиболее известный пользователям реляционной базы данных, несмотря на то, что он включает всего одну команду SELECT. Эта команда вместе со своими многочисленными опциями и предложениями используется для формирования запросов к реляционной базе данных.

Структура запроса на выборку:

SELECT[ALL|DISTINCT|DISTINCTROW][TOP n PERCENT]] <*список выбора*>
FROM <*список таблиц (запросов)* >
[WHERE <*условие отбора строк*>]
[GROUP BY <*список полей, по которым будут группироваться записи*>]
[HAVING <*условие отбора групп записей*>]
[ORDER BY <*определение сортировки записей*>];

SELECT – команда, определяющая запрос на выборку. Список выбора (список полей) содержит те поля, которые должны быть включены в результирующую таблицу запроса и их имена в этой новой таблице. Имена полей отделяются друг от друга запятыми. Имя поля формируется из имени таблицы и имени столбца, разделенных точкой.

Необязательные предикаты ALL, DISTINCT, DISTINCTROW определяют способ отбора строк. При использовании ALL в результирующую таблицу включаются все строки, удовлетворяющие указанным далее условиям. Предикат DISTINCT исключает повторяющиеся строки, основываясь на данных результирующего набора записей. Предикат DISTINCTROW исключает повторяющиеся строки, основываясь на данных полных строк исходной таблицы, независимо от того, включены ли в запрос те поля таблицы, данные которых различаются.

Необязательные предикат TOP n [PERCENT] возвращает первые *n* записей или *n* процентов набора записей, удовлетворяющих запросу.

Предложение FROM определяет таблицы или запросы, которые служат источником данных. Синтаксис предложение FROM :

FROM <имя таблицы (имена таблиц)>
[{INNER/ LEFT/ RIGHT }JOIN < связанная таблица>
ON <условие объединения >]

Если источником служит не текущая база данных, необходимо использовать предложение IN, чтобы задать путь к базе данных:

IN <'имя БД-источника' >.

Предложение FROM позволяет определить псевдоним таблицы, который можно использовать вместо полного имени таблицы при задании имен столбцов в списке выбора: FROM <имя таблицы > [AS <псевдоним>].

INNER JOIN – внутреннее объединение таблиц; LEFT JOIN – левое внешнее объединение; RIGHT JOIN – правое внешнее объединение.

Если имя таблицы (поля) совпадает с зарезервированным словом языка SQL (например, "ORDER") или содержит пробелы, то его необходимо заключить в квадратные скобки.

Предложение WHERE задает условие отбора строк. Для написания условий отбора можно использовать операторы сравнения, логические операторы, операторы LIKE, BETWEEN, IN, IS (*Таблица 1*).

Если необходимо результат выполнения одного запроса использовать в качестве данных для другого запроса, то первый будет являться подчиненным по отношению ко второму. Структура подчиненного запроса аналогична структуре внешнего запроса. Подчиненный запрос помещается в команду WHERE и заключается в круглые скобки.

Предложение ORDER BY задает порядок сортировки записей созданного запросом набора строк: по возрастанию (ASC) или по убыванию (DESC). По умолчанию принимается значение ASC:

Таблица 1

Категория операторов	Оператор	Пример	Описание
1	2	3	4
Арифмети-	+	[Итог]+[Надбавка]	Складывает два операнда
ческие операторы	-	Date() – 7	Считает разность двух операндов
	*	[Коробок]*[Цена]	Перемножает два операнда
	/	[Количество] / 5.33	Делит один операнд на другой
	\	[Количество] \ 2	Делит один целый операнд на другой нацело. При этом операнды с десятичными дробями округляются до целого
	Mod	[Коробок] Mod 3	Возвращает остаток отделения нацело
	^	[A] ^ [b]	Возводит операнд A в степень b
Операторы сравнивания[1]	<	1 < 100	Меньше
	<=	[a] <= 2	Меньше либо равно
	=	[b] = 3	Равно
	>	[a] > 2	Больше
	>=	[b] >= 3	Больше либо равно
	<>	1<>100	Неравно
Логические операторы[2]	And	[A] And [B]	Конъюнкция (логическое И)
	Or	[A] Or [B]	Дизъюнкция (логическое Или)
	Not	Not [A]	Логическое отрицание
	Xor	[A] Xor [B]	Исключающее ИЛИ
	Eqv	[A] Eqv [B]	Логическая эквивалентность
	Imp	[A] Imp [B]	Логическая импликация
Операторы конкатенации	+	“Visual “+”Basic”	Объединение двух текстовых значений в единую строку. Оба операнда эквивалентны, но & - предпочтительнее
	&	“Visual “ & ”Basic”	
Операторы идентификации	!	КлассОбъекта! ИмяОбъекта	Разделитель в ссылках на объекты
	.	ИмяОбъекта.Метод	Разделитель в ссылках на методы и свойства объектов
Операторы сравнения с образцом	Between	Between(-100) And (100)	Определяет, находится ли числовое значение в определенном диапазоне
	Is	[Имя] Is Null	Используется только для сравнения со значением. В примере выражение принимает истинное значение, когда в поле Имя нет никаких данных.

	In	In (“Москва”, ”Киев”, ”Мурманск”)	Определяет, является ли строковое значение элементом списка
			Значений
	Like	Like “Ив*” Like “db??”	Определяет, начинается ли строковое значение с указанных символов (символ “*” замещает любое число знаков, “?” – замещает только один знак)