

## Лабораторная работа №5

**ТЕМА:** Знакомство с основами языка SQL.

**ЦЕЛЬ:** Научить создавать запросы с использованием основных типов команд языка SQL.

**ПЛАН ЗАНЯТИЯ:**

- I.** Общие сведения о языке SQL.
- II.** Типы команд SQL.
- III.** Создание БД «фирма» и занесение исходных данных.
- IV.** Использование инструкции SELECT в языке запросов.
- V.** Выбор в SQL-запросе определенных полей и записей
- VI.** Использование в SQL-запросе сложных выражений и функций для вывода определенных записей.
- VII.** Выбор данных из более чем одной таблицы.
- VIII.** Вывод данных в определенном порядке.
- IX.** Агрегирующие функции в инструкции SELECT.
- X.** Группировка данных в инструкции SELECT.

### **I. ОБЩИЕ СВЕДЕНИЯ О ЯЗЫКЕ SQL**

SQL — это язык структурированных запросов (Structured Query Language) к реляционным базам данных. История SQL тесно связана с историей развития реляционных баз данных. В 1974-1975 годах был создан первый прототип реляционной СУБД. Кроме разработки самой СУБД, в рамках проекта System/R в компании IBM проводилась работа над созданием языка запросов базам данных. Первый язык запросов получил название SEQUEL — Structured English Query Language. Вторая реализация проекта System/R была установлена на компьютерах нескольких заказчиков IBM с целью опытной эксплуатации в 1978-1979 гг. В этой реализации язык запросов имел уже название SQL, но до сих пор еще можно услышать от пользователей старое название того языка.

В 1982 году компания IBM начала поставки на рынок коммерческого продукта SQL/Data System, а в 1983 году объявила о создании версии SQL/Data System для операционной системы VM/CMS. В 1983 году IBM выпустила новую реляционную СУБД — Database 2 (DB2). Благодаря влиянию компании IBM на рынок вычислительных систем SQL DB2 стал фактическим стандартом рынка баз данных.

Во второй половине восьмидесятых годов резко увеличилась производительность реляционных СУБД и с развитием реляционных технологий связывались большие надежды. Появились, в частности, новые версии СУБД Ingress и Oracle с производительностью, в два-три раза превышающей производительность предыдущих версий. Росту производительности СУБД способствовало и увеличение общего быстродействия компьютеров.

Опубликование в 1986 году стандарта SQL (ANSI/ISO) официально утвердило SQL как стандартный язык реляционных СУБД. С появлением более мощных персональных компьютеров и объединением их в локальные сети возникла необходимость в новых СУБД. Поставщики таких СУБД снова стали ориентироваться на SQL-технологии. И, наконец, SQL стал ключевой частью архитектуры клиент/сервер.

---

Первоначально SQL фирмы IBM имел простой синтаксис, но на протяжении нескольких лет производители программного обеспечения в области систем управления базами данных добавляли к своим реализациям новые возможности, многие из которых вошли в стандарты ANSI SQL. Версия языка ANSI SQL была принята организацией International Standards Organization (ISO), отделением ООН со штаб-квартирой в Женеве и комитетом International Electrotechnical Commission (IEC) как продукт ISO/IEC9075:1992 или *язык баз данных SQL (SQL-92)*. Отдельный стандарт ANSI X.3.168-1989 определяет *встроенный язык баз данных SQL (SQL-89)*. Современные СУБД поддерживают SQL-89 и многие дополнения из SQL-92. Кроме того, большинство СУБД имеют свои ключевые SQL-слова для создания патентованных диалектов SQL, таких как Transact-SQL (сервера SQL Server) или SQL Jet.

Историческое название SQL не совсем точно отражает суть этого языка, поскольку он давно уже предназначен не только для построения запросов (если, конечно, под «запросом» понимать только извлечение информации из базы данных). Сейчас SQL позволяет реализовать все функции СУБД:

- организацию данных,
- извлечение информации,
- модификацию данных,
- управление доступом,
- совместное использование данных,
- обеспечение целостности данных

В Visual Basic язык SQL применяется в основном для выполнения запросов, хотя очень удобно также удалять, добавлять и обновлять записи баз данных. Используя SQL-запросы, можно выбирать из таблиц базы данных только необходимые записи. При этом мы можем получить доступ не просто к одной таблице, а к сложной выборке из связанных между собой таблиц или наборов данных.

SQL-запросы можно также применять в приложениях, использующих объектные модели DAO, RDO1 или ADO. Кроме того, SQL как стандартный способ управления базами данных реализован во многих СУБД, включая Microsoft Access и SQL Server.

## II. ТИПЫ КОМАНД SQL

В ANSI SQL имеется шесть основных типов команд:

1. Команды *языка определения данных* (data definition language — DDL) позволяют создавать новые таблицы в базе данных, добавлять индексы и т.д. Основные команды языка определения данных приведены в таблице 1:

**Таблица 1.** Команды языка определения данных

CREATE TABLE	Создать таблицу
ALTER TABLE	Модифицировать таблицу
DROP TABLE	Удалить таблицу
CREATE INDEX	Создать индекс
ALTER INDEX	Модифицировать индекс
DROP INDEX	Удалить индекс

2. Команды *языка обработки данных* (data manipulation language — DML) используются для добавления, корректировки и удаления строк в таблицах и включают команды (см. таблицу 2):

**Таблица 2.** Команды языка обработки данных

INSERT	Вставить данные в таблицу
UPDATE	Обновить данные
DELETE	Удалить данные

3. Команда *языка запросов данных* (data query language — DQL) (единственная команда) используется для получения данных из таблиц и определения формы представления этих данных:

**Таблица 3.** Команда языка запросов данных

SELECT	Выполнить запрос из таблиц базы
--------	---------------------------------

4. Команды *языка управления данными* (data control language — DCL) определяют доступ отдельных пользователей и групп пользователей к объектам базы данных посредством полномочий, предоставляемых и отменяемых командами:

**Таблица 4.** Команды языка управления данными

GRANT	Предоставить привилегии
REVOKE	Отменить привилегии

5. Команды *языка обработки транзакций* (transaction processing language — TPL) обеспечивают обновление всех строк, используемых в операторе DML, и включают следующие команды:

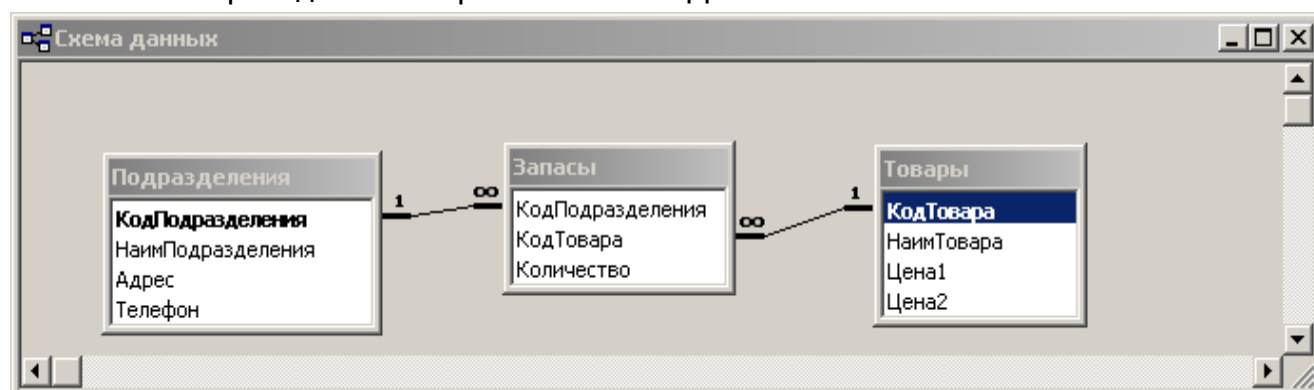
**Таблица 5.** Команды языка обработки транзакций

BEGIN TRANSACTION	Начать транзакцию
COMIT TRANSACTION	Завершить транзакцию
SAVE TRANSACTION	Создать точку сохранения внутри транзакции

6. Команды *языка управления курсором* (cursor control language — CCL) выполняют операции с отдельными строками одной или нескольких таблиц и включают команды: **DECLARE CURSOR, FETCH INTO И UPDATE WHERE CURRENT.**

### III. СОЗДАНИЕ БД «ФИРМА» И ЗАНЕСЕНИЕ ИСХОДНЫХ ДАННЫХ

Для выполнения лабораторной работы создайте базу данных «ФИРМА», в соответствии с приведенной на рис. 1 схемой БД.

**Рис. 1.** Схема базы данных *фирма.mdb*

Названия полей и типы данных приведены в таблице 6.

**Таблица 6. Типы полей БД «ФИРМА»**

Таблица	Поле	Тип и длина поля
Подразделения	КодПодразделения	Текстовое(4)
Подразделения	НаимПодразделения	Текстовое(50)
Подразделения	Адрес	Текстовое(60)
Подразделения	Телефон	Текстовое(10)
Запасы	КодПодразделения	Текстовое(4)
Запасы	КодТовара	Текстовое(12)
Запасы	Количество	Длинное Целое (Long Integer)
Товары	КодТовара	Текстовое(12)
Товары	НаимТовара	Текстовое(50)
Товары	Цена1	Денежный
Товары	Цена2	Денежный

Заполните созданные таблицы данными, приведенными на рис. 2-4

	КодПодраздел	НаимПодразделения	Адрес	Телефон
+	0429	СКЛАД (Мультимедиа)	ул. П.И. Чайковского, 22	333-33-05
+	1506	СКЛАД (МНЕВНИКИ)	ул. Советская, 77	444-33-45
+	1518	СКЛАД (НЕЙРОН)	пр. Петровича	324-55-45
+	1958	СКЛАД (Брак)	пер. Симоновича	334-33-41
*				

Запись: 1 из 4

**Рис. 2. Содержимое таблицы «Подразделения»**

	КодТовара	НаимТовара	Цена1	Цена2
+	006019165803	(S) Road Rash II	42,00p.	49,00p.
+	006019170049	(S) Tazmania	30,00p.	35,00p.
+	010414114438	(VCD) Патриот	138,00p.	161,00p.
+	010415090732	(VCD) Пастырь	111,00p.	130,00p.
+	010415090937	(VCD) Шоссе в никуда	120,00p.	140,00p.
+	010428144348	(VCD) Доберман	135,00p.	158,00p.
+	028019095823	(VCD) Diber Marouani	90,00p.	105,00p.
+	028019100018	(VCD) Wham! The best of	105,00p.	123,00p.
+	028019100222	(VCD) Голая мишень	90,00p.	105,00p.
+	036019164741	(S) Cool Spot	42,00p.	49,00p.
+	174500175359	(3DC) Street Fighter 3 Wir	750,00p.	875,00p.
+	174500175521	(3DC) Star Gladiator 2	750,00p.	875,00p.
+	186100175232	(3DC) Super Runabout	750,00p.	875,00p.
*			0,00p.	0,00p.

Запись: 1 из 13

**Рис. 3. Содержимое таблицы «Товары»**

КодПодраздел	КодТовара	Количество
0429	006019165803	3000
1506	006019165803	200
1518	006019165803	1200
0429	006019170049	1000
1506	006019170049	500
1518	006019170049	1000
1506	010414114438	400
1958	010414114438	3300
1506	010428144348	800
1958	010428144348	4200
0429	036019164741	2000
1506	036019164741	600
1518	036019164741	1400
1506	174500175359	600
1958	174500175359	2200
0429	174500175359	1500
1958	174500175359	2300
0429	186100175232	2000
1958	186100175232	1300
*		0

Рис. 4. Содержимое таблицы «Запасы»

#### IV. ИСПОЛЬЗОВАНИЕ ИНСТРУКЦИИ SELECT В ЯЗЫКЕ ЗАПРОСОВ

Для работы с базами данных в Visual Basic применяется язык запросов, включающий единственную инструкцию SELECT. Рассмотрим некоторые вопросы, связанные с ее использованием. Синтаксис (неполный) инструкции SELECT в SQL Jet следующий:

```
SELECT [ALL | DISTINCT | DISTINCTROW | TOP] { * | table.* |
[table.] field1 [AS alias1 [, [table.] field2 [AS alias2] [, ...]]}
FROM table1 [table1Alias] [, table2 [table2Alias]] [, ...] [IN externaldatabase ]
[WHERE criteria]
[GROUP BY groupfieldlist]
[HAVING groupcriteria]
[ORDER BY field1 [ASC | DESC ], field2 [ASC | DESC ][, ...]]
[WITH OWNERACCESS OPTION]
```

Инструкция **SELECT** включает следующие основные элементы:

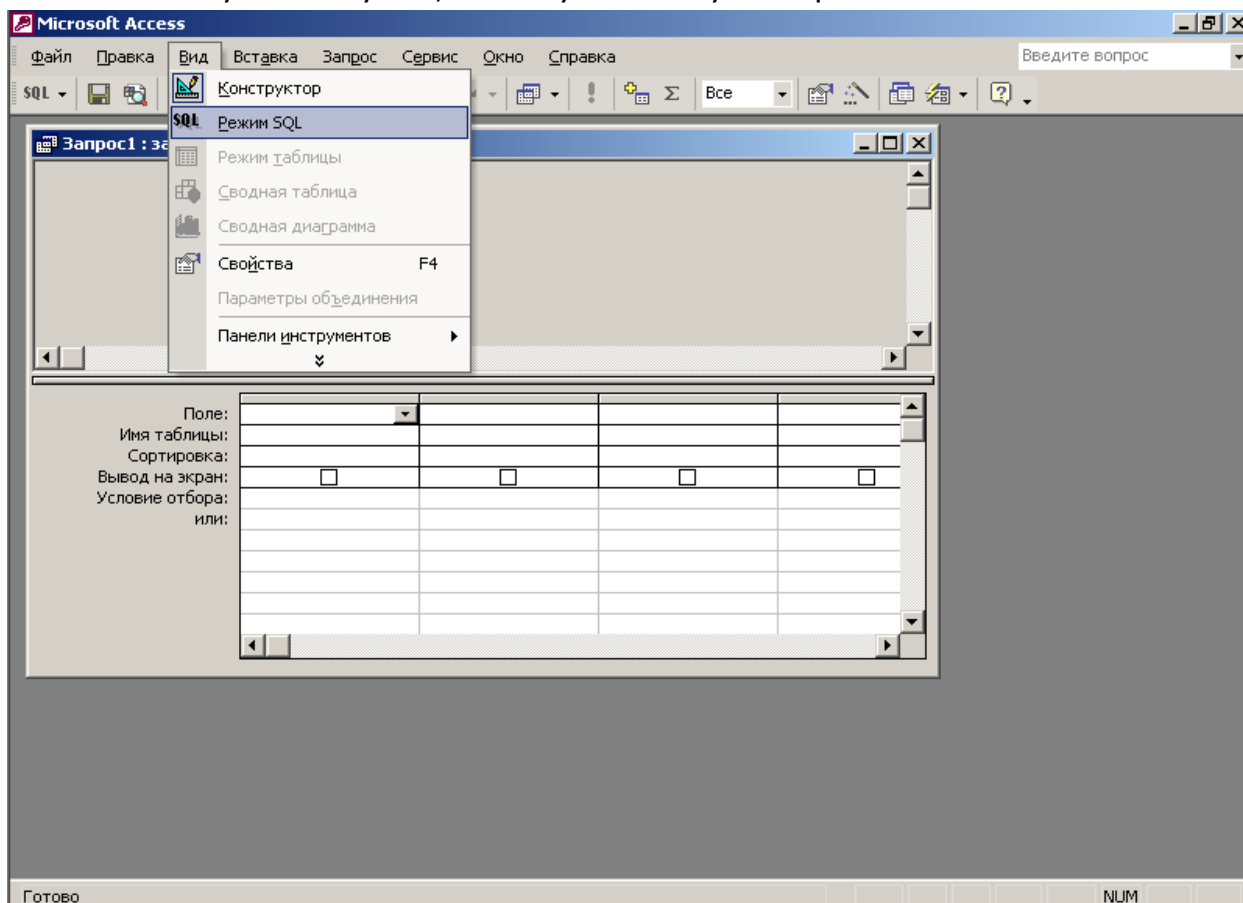
- **SELECT** означает, что из некоторых таблиц базы данных необходимо выбрать набор (таблицу данных).
- Необязательные слова **ALL**, **DISTINCT**, **DISTINCTROW** и **TOP** называются *предикатами* (predicates) и определяют выбор следующим образом: **ALL** указывает, что в набор передаются все строки (даже с дублируемыми значениями); **DISTINCT** указывает, что в набор передаются только недублированные строки; **DISTINCTROW** указывает, что в результирующий набор будет включена каждая строка, в которой есть отличие в значении любого из полей исходных таблиц (а не только полей, указанных для отображения

в операторе **SELECT**); **TOP** используется для отображения некоторого количества (точного или в процентном отношении) начальных или конечных записей из результирующего набора.

- Список { \* | **table.\*** | [**table.**]field1 [**AS alias1**] [, [**table.**]field2 [**AS alias2**] [, ...]] } (фигурные скобки здесь обозначают список) состоит из имен полей таблиц(ы) запроса. Звездочка (\*) означает выбор всех полей таблицы. Если в запросе указывается несколько таблиц, то для определения поля используется наименование таблицы, отделяемое от имени поля точкой (.). Поле может получать «алиасное» имя при помощи ключевого слова **As**.
- После слова **FROM** указываются таблицы, из которых выбираются ранее указанные поля. Здесь **table1** (**,table2**) — это имя таблицы (или таблиц), содержащей данные, **externaldatabase** — имя базы данных, если не используется текущая база.

**ПРИМЕЧАНИЕ:** Нет никаких правил написания SQL-инструкций относительно их положения в строке. Можно писать инструкцию в одной строке, можно — в нескольких.

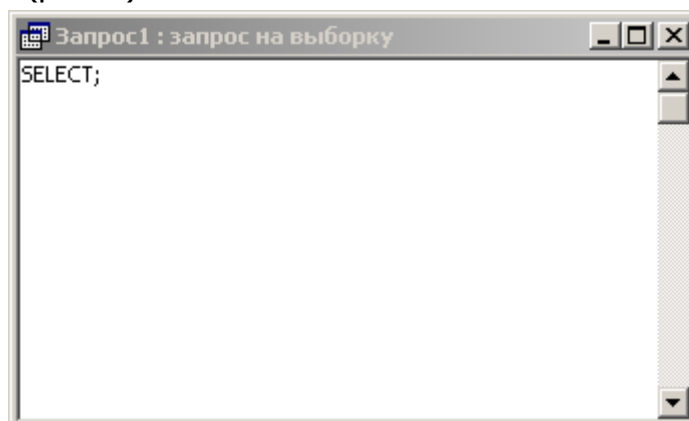
Для тестирования SQL-инструкций в среде Access выберите в левой части главного окна Access в меню **Объекты** опцию **Запросы** и дважды щелкните команду **Создание запроса в режиме конструктора**<sup>1</sup>. Появившееся окно Добавление таблицы нам в данном случае не нужно, поэтому его следует закрыть.



**Рис. 5.** Режим SQL меню **Вид**

<sup>1</sup> Это не единственный способ создания SQL запроса

Не обращая внимания на средства Access, предназначенные для «легкого» создания запроса (в нижней части диалогового окна), выберите команду **Режим SQL** в меню **Вид** (рис. 5). Чтобы, получить доступ к тому окну, в котором можно будет набирать SQL-инструкции (рис. 6).



**Рис. 6.** Окно для создания SQL-инструкций

Инструкцию SELECT будем изучать по принципу «от простого — к сложному»: сначала выбросим из полного синтаксиса этой инструкции все необязательные элементы, а затем постепенно будем использовать их, получая более сложные запросы. Если отбросить все необязательные предложения из списка { \* | **table.\*** | **[table.]field1 [AS alias1] [, [table.]field2 [AS alias2] [, ...]]** } и оставить только элемент \*, то синтаксис самой простой SELECT-инструкции (SQL-запроса) будет иметь вид:

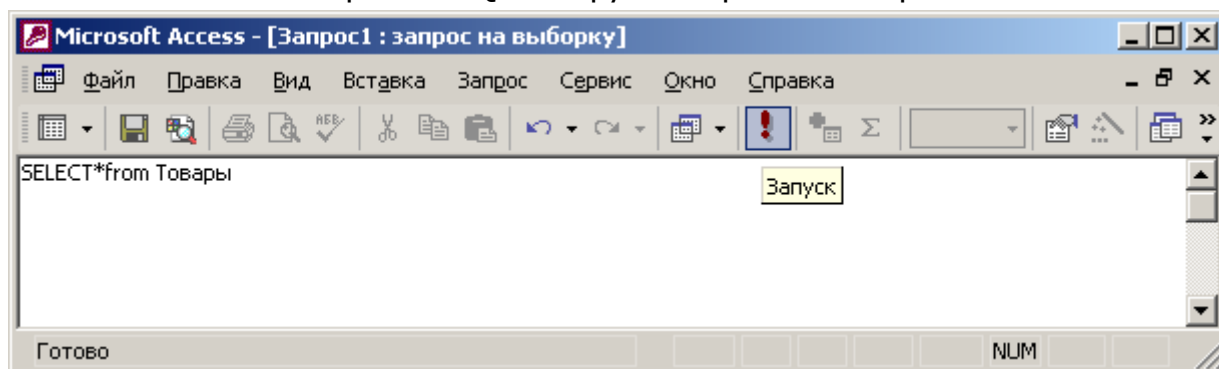
**SELECT \* FROM table**

Здесь знак \* означает выбор значений всех полей, а **table** — имя таблицы, из которой выбираются значения.

Поэтому самая простая SELECT-инструкция для одной из рассматриваемых нами таблиц может выглядеть так:

**SELECT \* FROM Товары**

Введите эту инструкцию в окно **Запрос 1: запрос на выборку**, как показано на рис. 7, и щелкните кнопку **Запуск** на панели **Конструктор запросов**. Результат выполнения этой очень простой SQL-инструкции приведен на рис. 8.



**Рис. 7.** Окно для ввода инструкций SQL

КодТовара	НаимТовара	Цена1	Цена2
006019165803	(S) Road Rash II	42,00р.	49,00р.
006019170049	(S) Tazmania	30,00р.	35,00р.
010414114438	(VCD) Патриот	138,00р.	161,00р.
010415090732	(VCD) Пастырь	111,00р.	130,00р.
010415090937	(VCD) Шоссе в никуда	120,00р.	140,00р.
010428144348	(VCD) Доберман	135,00р.	158,00р.
028019095823	(VCD) Diber Marouani	90,00р.	105,00р.
028019100018	(VCD) Wham! The best of	105,00р.	123,00р.
028019100222	(VCD) Голая мишень	90,00р.	105,00р.
036019164741	(S) Cool Spot	42,00р.	49,00р.
174500175359	(3DC) Street Fighter 3 Wir	750,00р.	875,00р.
174500175521	(3DC) Star Gladiator 2	750,00р.	875,00р.
186100175232	(3DC) Super Runabout	750,00р.	875,00р.
*		0,00р.	0,00р.

Рис. 8. Результат выполнения SQL инструкции в среде MS Access

## V. ВЫБОР В SQL-ЗАПРОСЕ ОПРЕДЕЛЕННЫХ ПОЛЕЙ И ЗАПИСЕЙ

Чаще всего вам не нужно будет получать набор со всеми полями одной (или более) таблицы. Например, поскольку поле **КодТовара** в таблице **ТОВАРЫ** предназначено в основном для связи с другими таблицами, то не часто его следует включать в SQL-запрос. В SELECT-инструкции можно указать конкретные наименования используемых полей некоторой таблицы, используя следующий синтаксис:

```
SELECT field1 [,field2 [, ...]]
FROM table
```

Например:

```
SELECT НаимТовара, Цена1
FROM Товары
```

Результат этого запроса представлен на рис. 9.

НаимТовара	Цена1
(S) Road Rash II	42,00р.
(S) Tazmania	30,00р.
(VCD) Патриот	138,00р.
(VCD) Пастырь	111,00р.
(VCD) Шоссе в никуда	120,00р.
(VCD) Доберман	135,00р.
(VCD) Diber Marouani	90,00р.
(VCD) Wham! The best of	105,00р.
(VCD) Голая мишень	90,00р.
(S) Cool Spot	42,00р.
(3DC) Street Fighter 3 Wir	750,00р.
(3DC) Star Gladiator 2	750,00р.
(3DC) Super Runabout	750,00р.
*	0,00р.

Рис. 9. Пример использования SELECT-инструкции



Для выбираемых в запросе полей можно указать отличные от наименований полей заголовки. Чтобы указать наименования полей результирующего запроса, необходимо использовать такой синтаксис:

```
SELECT field1 [AS alias1] [,field2 [AS alias2] [, . ]]
FROM table
```

В следующей инструкции исправим наименования в запросе (результат выполнения — на рис. 10):

```
SELECT НаимТовара as [Наименование товара], Цена1 as [Цена закупочная]
FROM Товары
```

Наименование товара	Цена закупочная
(S) Road Rash II	42,00p.
(S) Tasmania	30,00p.
(VCD) Патриот	138,00p.
(VCD) Пастырь	111,00p.
(VCD) Шоссе в никуда	120,00p.
(VCD) Доберман	135,00p.
(VCD) Diber Marouani	90,00p.
(VCD) Wham! The best of	105,00p.
(VCD) Голая мишень	90,00p.
(S) Cool Spot	42,00p.
(3DC) Street Fighter 3 Wir	750,00p.
(3DC) Star Gladiator 2	750,00p.
(3DC) Super Runabout	750,00p.
*	0,00p.

**Рис. 10.** Пример инструкции с указанием заголовков, отличных от наименований полей БД

Ключевое слово **WHERE** в синтаксисе инструкции **SELECT** позволяет указывать определенные типы записей, которые должны попадать в набор. При этой инструкция **SELECT** имеет следующий синтаксис:

```
SELECT { * | field1 [AS alias1] [,field2 [AS alias2]]}
FROM table
[WHERE criteria]
```

Например, можно запросить только те товары, цены которых не более 100 денежных единиц (результат выполнения — на рис. 11):

Наименование товара	Цена закупочная
(S) Road Rash II	42,00p.
(S) Tasmania	30,00p.
(VCD) Diber Marouani	90,00p.
(VCD) Голая мишень	90,00p.
(S) Cool Spot	42,00p.
*	0,00p.

**РИС. 11.** Использование ключевого слова **WHERE**

```
SELECT НаимТовара as [Наименование товара], Цена1 as [Цена закупочная]
FROM Товары
WHERE Цена1 < 100
```

## VI. ИСПОЛЬЗОВАНИЕ В SQL-ЗАПРОСЕ СЛОЖНЫХ ВЫРАЖЕНИЙ И ФУНКЦИЙ ДЛЯ ВЫВОДА ОПРЕДЕЛЕННЫХ ЗАПИСЕЙ

В области слова **WHERE** можно располагать довольно сложное условное выражение с использованием знаков логических операций и функций. Например:

```
SELECT НаимТовара as [Наименование товара], Цена1 as [Цена закупочная]
FROM Товары
WHERE Цена1 > 50 AND Цена1 < 200
```

или

```
SELECT НаимТовара as [Наименование товара], Цена1 as [Цена закупочная]
FROM Товары
WHERE Цена1 > 50 AND Len(НаимТовара) < 20
```

В первом случае запрашиваются товары, оптовая цена которых находится в диапазоне оптовых цен (55-200), а во втором — цены которых больше 50 денежных единиц и длина наименования не превышает 20 символов. Выполните данные запросы самостоятельно.

Кроме операций **<** и **>**, в инструкции **SELECT** можно использовать операции **=** (равно), **<=** (меньше или равно) и **>=** (больше или равно), а также **AND**, **OR** и **NOT**. Еще большие возможности по отбору необходимых записей предоставляют такие операторы, как **IS NULL**, **BETWEEN**, **IN** и **LIKE**.

Оператор **IS NULL** позволяет найти в таблице записи, в полях которых не указаны данные, например:

```
SELECT НаимТовара as [Наименование товара], Цена1 as [Цена закупочная]
FROM Товары WHERE Цена2 IS NULL OR Цена1 < 50
```

Оператор **BETWEEN** позволяет указать диапазон, в котором находятся данные некоторого поля, например:

```
SELECT НаимТовара as [Наименование товара], Цена1 as [Цена закупочная]
FROM Товары
WHERE Цена1 BETWEEN 42 AND 120
```

Оператор **IN** позволяет указать список, в котором находятся данные некоторого поля, например:

```
SELECT НаимТовара as [Наименование товара], Цена1 as [Цена закупочная]
FROM Товары
WHERE Цена1 IN (42, 105, 750)
```

В SQL-инструкции можно также использовать оператор **LIKE**, например инструкция:

```
SELECT НаимТовара as [Наименование товара], Цена1 as [Цена закупочная]
FROM Товары
WHERE НаимТовара LIKE "(VCD)*"
```

позволяет получить только те записи (рис. 12), у которых в наименовании первые пять символов совпадают со строкой "(VCD)".

Наименование товара	Цена закупочная
(VCD) Патриот	138,00р.
(VCD) Пастырь	111,00р.
(VCD) Шоссе в никуда	120,00р.
(VCD) Доберман	135,00р.
(VCD) Diber Marouani	90,00р.
(VCD) Wham! The best of	105,00р.
(VCD) Голая мишень	90,00р.
*	0,00р.

**Рис. 12.** Записи, у которых в наименовании первые пять символов совпадают со строкой "(VCD)"

Оператор **LIKE** можно использовать для контекстного поиска, например, если ввести текстовую строку, содержащую часть наименования товара, то эту строку легко использовать в виде шаблона. Следующая инструкция выполняет поиск записей, в которых наименование содержит в качестве подстроки строку "мишень" (результат — на рис. 13):

```
SELECT НаимТовара as [Наименование товара], Цена1 as [Цена закупочная]
FROM Товары
WHERE НаимТовара LIKE "*мишень*"
```

Наименование товара	Цена закупочная
(VCD) Голая мишень	90,00р.
*	0,00р.

**Рис. 13.** Поиск записей, наименование которых содержит в качестве подстроки строку «мишень»

Оператор **NOT**, который инвертирует логическое выражение, может использоваться с операторами **IS NULL**, **BETWEEN**, **IN**, **LIKE**. При этом, практически, речь идет об операторах **IS NOT NULL**, **NOT BETWEEN**, **NOT IN** и **NOT LIKE**, например:

```
SELECT НаимТовара as [Наименование товара], Цена1 as [Цена закупочная]
FROM Товары
WHERE Цена2 IS NOT NULL OR Цена1 < 5
```

```
SELECT НаимТовара as [Наименование товара], Цена1 as [Цена закупочная]
FROM Товары
WHERE Цена1 NOT BETWEEN 42 AND 120
```

```
SELECT НаимТовара as [Наименование товара], Цена1 as [Цена закупочная]
FROM Товары
WHERE Цена1 NOT IN (42, 105, 750)
```

```
SELECT НаимТовара as [Наименование товара], Цена1 as [Цена закупочная]
FROM Товары
WHERE НаимТовара NOT LIKE "(VCD) *"
```

## Функция Format

Результирующие данные запроса можно форматировать с использованием функции **Format**. В следующем запросе данные форматируются при помощи строки "### ##0.00\$" (результат — на рис. 14):

```
SELECT НаимТовара as [Наименование товара],
       Format(Цена1*1.2,"### ##0.00p") as [Цена оптовая]
FROM Товары
WHERE Цена1 < 100
```

Наименование товара	Цена закупочная
(S) Road Rash II	50,40p
(S) Tazmania	36,00p
(VCD) Diber Marouani	108,00p
(VCD) Голая мишень	108,00p
(S) Cool Spot	50,40p
*	

Рис. 14. Использование функции **Format**

Обратите внимание еще и на то, что функция **Format** в качестве первого аргумента получает произведение **Цена1\*1.2**. Таким образом, мы получаем другую цену из некоторой базовой.

Для форматирования выводимых в запросе данных можно использовать функции преобразования строк. Например, в следующем запросе наименования товаров выводятся символами верхнего регистра, поскольку здесь используется функция **StrConv** (результат — на рис. 15):

```
SELECT StrConv(НаимТовара,1) AS [Наименование товара], Цена1 AS [Цена за-  
купочная] FROM Товары
```

Наименование товара	Цена закупочная
(S) ROAD RASH II	42,00p.
(S) TAZMANIA	30,00p.
(VCD) ПАТРИОТ	138,00p.
(VCD) ПАСТЫРЬ	111,00p.
(VCD) ШОССЕ В НИКУДА	120,00p.
(VCD) ДОБЕРМАН	135,00p.
(VCD) DIBER MAROUANI	90,00p.
(VCD) WHAM! THE BEST OF	105,00p.
(VCD) ГОЛАЯ МИШЕНЬ	90,00p.
(S) COOL SPOT	42,00p.
(3DC) STREET FIGHTER 3 WIMPACT	750,00p.
(3DC) STAR GLADIATOR 2	750,00p.
(3DC) SUPER RUNABOUT	750,00p.
*	0,00p.

Рис. 15. Использование функции **StrConv**

## VII. ВЫБОР ДАННЫХ ИЗ БОЛЕЕ ЧЕМ ОДНОЙ ТАБЛИЦЫ

Имея базу данных из нескольких таблиц, мы до сих пор получали при помощи инструкции **SELECT** наборы данных только из одной таблицы. Выбор данных из нескольких таблиц — это не только необходимость, но и истинное удовольствие, причем не только для начинающих изучать язык SQL. Разработчикам Windows-приложений, использующим базы данных, очень часто приходится создавать сложные SQL-запросы. И чем сложнее построенный запрос, тем большее удовлетворение получает автор этого запроса.

Рассмотрим задачу выбора из базы данных товаров некоторого склада. Перед тем как рассмотреть использование слова **WHERE** для связи таблиц, заметим, что в инструкции **SELECT** можно перед именем поля указывать имя таблицы, которое отделяется от имени поля точкой:

```
SELECT { * | [table1.]field1 [AS alias1] [, [table2.]field2 [AS alias2] [, ...]] }
FROM table1 [table1Alias] [, table2 [table2Alias]] [, ...]
[WHERE criteria]
```

Например:

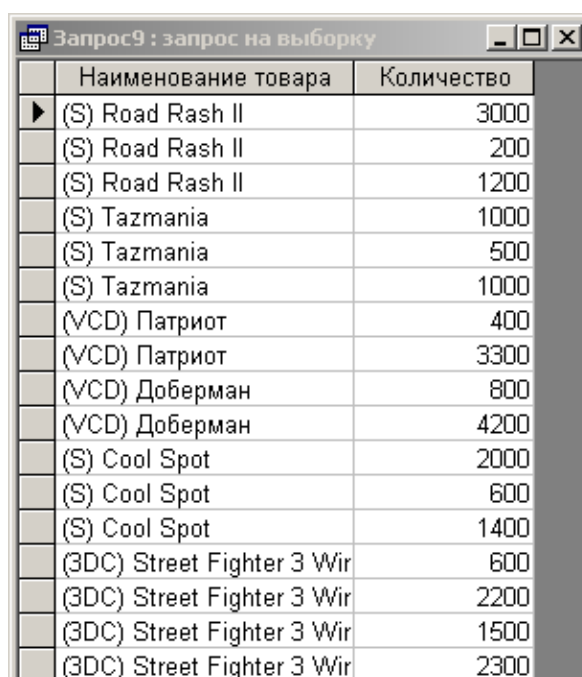
```
SELECT Товары.НаимТовара AS [Наименование товара],
Товары.Цена1 AS [Цена закупочная]
FROM Товары
```

или

```
SELECT а.НаимТовара AS [Наименование товара], а.Цена1 AS [Цена закупочная]
FROM Товары а
```

Во второй инструкции использовано альтернативное имя таблицы (локальный псевдоним), но обе инструкции выдают один и тот же результирующий набор.

Следующая инструкция позволяет получить наименования (из таблицы **ТОВАРЫ**) и количества (из таблицы **ЗАПАСЫ**) товаров (результат выполнения — на рис. 16):



Наименование товара	Количество
(S) Road Rash II	3000
(S) Road Rash II	200
(S) Road Rash II	1200
(S) Tazmania	1000
(S) Tazmania	500
(S) Tazmania	1000
(VCD) Патриот	400
(VCD) Патриот	3300
(VCD) Доберман	800
(VCD) Доберман	4200
(S) Cool Spot	2000
(S) Cool Spot	600
(S) Cool Spot	1400
(3DC) Street Fighter 3 Wir	600
(3DC) Street Fighter 3 Wir	2200
(3DC) Street Fighter 3 Wir	1500
(3DC) Street Fighter 3 Wir	2300

**Рис. 16.** Результат инструкции, позволяющей получить наименования (из таблицы **ТОВАРЫ**) и количества (из таблицы **ЗАПАСЫ**)

```
SELECT Товары.НаимТовара AS [Наименование товара], Запасы.Количество
AS [Количество]
FROM Товары, Запасы WHERE Товары.КодТовара= Запасы.КодТовара
```

Обратите внимание на выражение **Товары.КодТовара=Запасы.КодТовара**.

Именно оно используется для связи таблиц **ТОВАРЫ** и **ЗАПАСЫ** по ключевому полю **КодТовара**. Результат выполнения инструкции напоминает инвентаризационную ведомость без указания места, где находится товар.

Следующая инструкция позволяет получить наименования (из таблицы **ТОВАРЫ**) и количества (из таблицы **ЗАПАСЫ**) только для тех кодов товаров, записи которых в таблице **ЗАПАСЫ** в поле **КодПодразделения** содержат строку "0429":

```
SELECT Товары.НаимТовара AS [Наименование товара], Запасы.Количество
AS [Количество]
FROM Товары, Запасы
WHERE Запасы.КодПодразделения ='0429'
AND Товары.КодТовара= Запасы.КодТовара
```

При помощи локальных псевдонимов можно сократить предыдущую инструкцию (результат — на рис. 17):

```
SELECT а.НаимТовара AS [Наименование товара], b.Количество AS [Количество]
FROM Товары а, Запасы b
WHERE b.КодПодразделения ='0429' AND а.КодТовара=b.КодТовара
```

Наименование товара	Количество
(S) Road Rash II	3000
(S) Tasmania	1000
(S) Cool Spot	2000
(3DC) Street Fighter 3 Wimpact	1500
(3DC) Super Runabout	2000

**Рис. 17.** Наименования товаров только для одного склада

Коды как товаров, так и складов не предназначены для использования клиентами. Они необходимы для связи между таблицами. По этой причине следует избегать от «публичного» использования кода склада в SELECT-инструкции:

```
SELECT а.НаимТовара AS [Наименование товара], b.Количество AS [Количество]
FROM Товары а, Запасы b, Подразделения с
WHERE а.КодТовара=b.КодТовара
AND b.КодПодразделения =с.КодПодразделения
AND с.НаимПодразделения ="СКЛАД (Мультимедиа) "
```

В этой инструкции использована таблица **ПОДРАЗДЕЛЕНИЯ**, а при помощи поля **КодПодразделения** она связана с таблицей **ЗАПАСЫ**.

## VIII. ВЫВОД ДАННЫХ В ОПРЕДЕЛЕННОМ ПОРЯДКЕ

Для сортировки данных в инструкции **SELECT** имеются слова **ORDER BY**:

```
SELECT { * | [table.]field1 [AS alias1] [, [table.]field2 [AS alias2] [, ...]]
FROM table1 [table1Alias] [, table2 [table2Alias]] [, ...]
[WHERE criteria]
[ORDER BY field1 [ASC | DESC] [, field2 [ASC | DESC] [, ...]]]
```

Здесь к тем элементам SQL-запроса, которые уже рассмотрены, добавлено обязательное предложение ORDER BY. Как следует из синтаксиса инструкции SELECT, используя слова ASC и DESC, можно изменять «направление» сортировки («по возрастанию» и «по убыванию»). Сортировать можно по нескольким полям (сначала по одному, затем — по другому, и так далее) и даже по различным элементам одного и того же поля с использованием функций.

В следующей инструкции используется сортировка выводимого набора по наименованиям (результат — на рис. 18):

```
SELECT а.НаимТовара AS [Наименование товара], б.Количество AS [Количество]
FROM Товары а, Запасы б
WHERE а.КодТовара=б.КодТовара AND б.КодПодразделения='0429'
ORDER BY а.НаимТовара
```

Наименование товара	Количество
(3DC) Street Fighter 3 Wimpact	1500
(3DC) Super Runabout	2000
(S) Cool Spot	2000
(S) Road Rash II	3000
(S) Tazmania	1000

Рис. 18. Сортировка данных в инструкции **SELECT**

## IX. АГРЕГИРУЮЩИЕ ФУНКЦИИ В ИНСТРУКЦИИ SELECT

В инструкциях языка ANSI SQL предусмотрены так называемые агрегирующие функции, которые определяют количество записей, вычисляют суммы всех значений полей в наборе, находят минимальные или максимальные, а также средние значения. К агрегирующим функциям относятся функции **COUNT**, **SUM**, **MAX**, **MIN** и **AVG**.

Функция **COUNT** используется для определения количества записей в запросе. Например, в окне на рис. 19 введена инструкция для вычисления количества записей в таблице **ТОВАРЫ** и результат выполнения этой инструкции.

Рис. 19. Инструкция **SELECT** для определения общего количества записей в таблице и ее результат

## Задание

Самостоятельно выполните и сохраните следующие инструкции:

1. Вычислить количество записей, в которых значение поля **Цена1** меньше **100**:  

```
SELECT COUNT (НаимТовара) AS [Всего наименований]
FROM Товары
WHERE Цена1 < 100
```
2. Используя функцию **SUM** для группы строк вычислить итоговую сумму значений некоторого поля:



```
SELECT SUM(Количество) as [Общее количество товаров] FROM Запасы
```

При использовании функции **SUM** можно ограничить количество записей запроса при помощи некоторого условия:

```
SELECT SUM(Количество) AS [Количество на складе]
FROM Запасы
WHERE КодПодразделения='0429'
```

3. Используя функцию **AVG**, которая позволяет найти среднее значение для строк, входящих в запрос, определить среднюю цену на товары, наименования которых начинаются со строки "(VCD)":

```
SELECT Format(AVG(Цена1), "### ##0.00p") AS [Среднее значение]
FROM Товары
WHERE НаимТовара LIKE '(VCD)*'
```

4. Используя функции **MAX** и **MIN** определить максимальное значение цены на товары, наименования которых начинаются со строки "**(VCD)**":

```
SELECT Format(MAX(Цена1), "### ##0.00p") AS [Максимальная цена]
FROM Товары
WHERE НаимТовара LIKE '(VCD)*'
```

и минимальное значение цены на товары, наименования которых начинаются со строки "**(VCD)**":

```
SELECT Format(MIN(Цена1), "### ##0.00p") AS [Минимальная цена]
FROM Товары
WHERE НаимТовара LIKE '(VCD)*'
```

## Х. ГРУППИРОВКА ДАННЫХ В ИНСТРУКЦИИ SELECT

Часто в наборах, получаемых при помощи **SELECT**-инструкций, встречаются повторяющиеся значения. Для рассматриваемой нами базы данных это могут быть, например, наименования товаров в запросе с инвентаризационной ведомостью, номера и даты накладных в таблице со спецификациями накладных. Такие повторяющиеся значения можно объединять в группы, используя для групп агрегирующие функции.

В инструкции **SELECT** для объединения значений в группы используется предложение **GROUP BY**.

Следующая инструкция имеет результатом список складов и суммарные количества товаров на каждом складе (результат — на рис. 20):

```
SELECT с.НаимПодразделения, SUM(b.Количество) AS Количество
FROM Запасы b, Подразделения с
WHERE b.КодПодразделения=с.КодПодразделения
GROUP BY с.НаимПодразделения
```

НаимПодразделения	Количество
СКЛАД (Брак)	13300
СКЛАД (МНЕВНИКИ)	3100
СКЛАД (Мультимедиа)	9500
СКЛАД (НЕЙРОН)	3600

**Рис. 20.** Список складов и суммарные количества товаров на каждом складе



Эту же инструкцию можно выполнить в упрощенном варианте только для одной таблицы, используя вместо наименований складов их коды:

```
SELECT b.КодПодразделения, SUM(b.Количество) AS Количество
FROM Запасы b
GROUP BY b.КодПодразделения
```

Здесь суммируются значения поля **КОЛИЧЕСТВО** для одинаковых значений полей **КодПодразделения**. В этом — смысл предложения **GROUP BY**. В предыдущем запросе мы просто вместо кодов подставили (посредством **WHERE b.КодПодразделения = c.КодПодразделения**) наименования складов, используя для этого справочник **Подразделения**; смысл самого запроса, практически, такой же.

Предположим, что нам нужна информация о суммарном количестве товара только на тех складах, где это количество меньше определенного числа. Чтобы выполнить этот запрос, мы не можем использовать предложение **WHERE**, так как оно «работает» со значениями данных, расположенными в таблице, а нам нужно наложить некоторые ограничения на вычисляемые данные. Для решения подобной задачи следует использовать предложение **HAVING**, которое так же связано с предложением **GROUP BY**, как **WHERE** с **SELECT**. Другими словами, предложение **HAVING** налагает некоторые условия на выбранные посредством предложения **GROUP BY** данные.

Следующая инструкция позволяет получить список складов и суммарное количество товаров на них. Причем в список включаются только те склады, на которых суммарное количество товаров меньше чем **10000** (результат — на рис. 21):

```
SELECT c.НаимПодразделения, SUM(b.Количество) AS Количество
FROM Запасы b, Подразделения c
WHERE b.КодПодразделения = c.КодПодразделения
GROUP BY c.НаимПодразделения
HAVING SUM(b.Количество) < 10000
```

НаимПодразделения	Количество
СКЛАД (МНЕВНИКИ)	3100
СКЛАД (Мультимедиа)	9500
СКЛАД (НЕЙРОН)	3600

**Рис. 21.** Список складов, на которых суммарное количество товаров меньше чем **10000**

## XI. КОНТРОЛЬНЫЕ ВОПРОСЫ:

1. Раскройте понятие SQL.
2. Какие основные функции СУБД позволяет реализовать SQL?
3. Перечислите основные типы команд SQL.
4. Приведите общий синтаксис инструкции SELECT.
5. Как в SQL запросе выбрать определенное поле? Запись?
6. Какие операторы используются при создании сложных выражений и функций?
7. Какие имеются способы форматирования данных запроса?
8. Выбор данных из нескольких таблиц.
9. Что такое агрегирующие функции? Приведите примеры их использования.
10. Назначение, применение и синтаксис команды GROUP BY.