Тема 7

Запросы

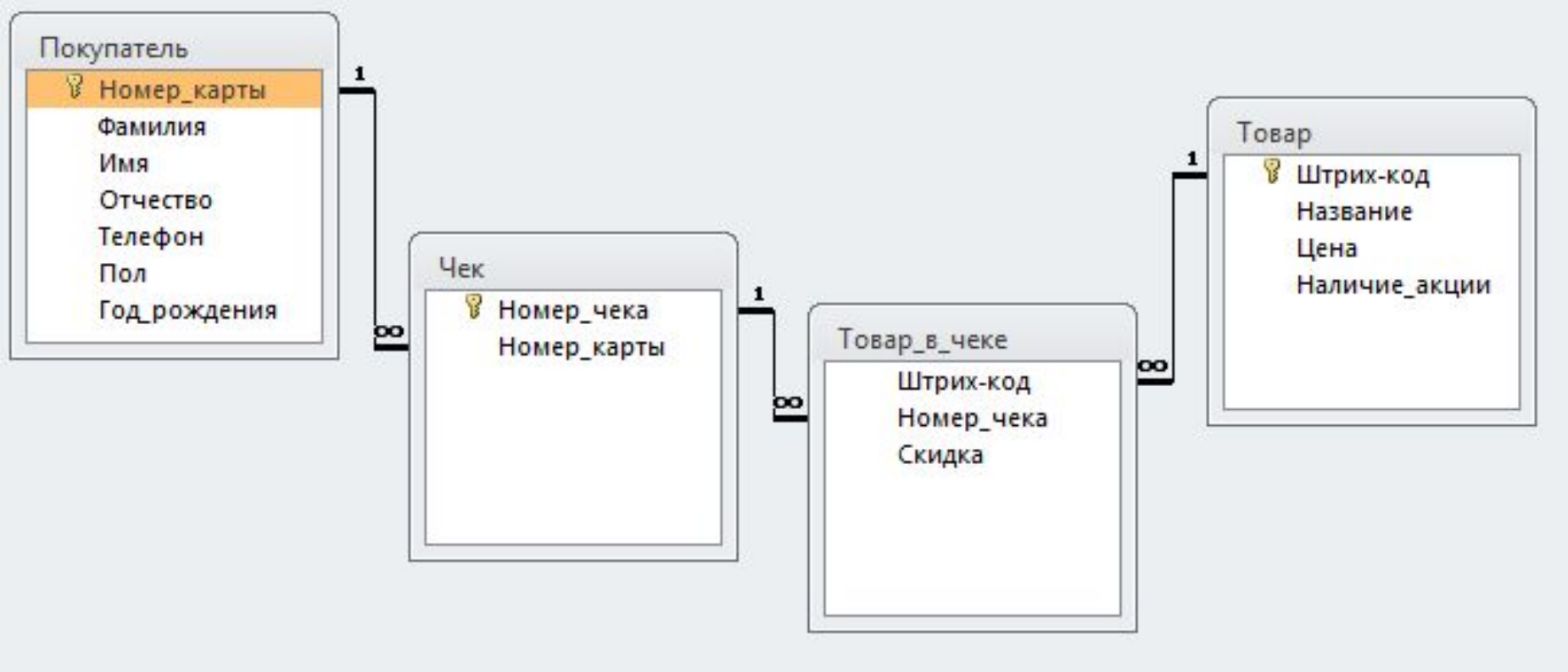
Язык SQL

Язык структурированных запросов Structured Query Language (SQL) является стандартным языком обработки данных, используемым большинством СУБД.

Язык был предложен исследовательской лабораторией фирмы IBM в начале 1970-х годов для реализации реляционной модели данных Э. Ф.Кодда (E.F.Codd).

Язык SQL был разработан задолго до появления графических интерфейсов пользователя, поэтому он ориентирован на использование текста для написания команд. Использование SQL позволяет выполнять более гибкую обработку данных, например, путем встраивания команд SQL в текст программ на каком-либо языке программирования. С помощью SQL можно определять структуры данных, а также запрашивать и обновлять информацию в базе данных. Совокупность команд, служащих для определения данных, называют ***языком определения данных*** (Data Description Language, DDL), а совокупность команд для обновления и запроса данных – ***языком манипулирования данных*** (Data Manipulation Language, DML).

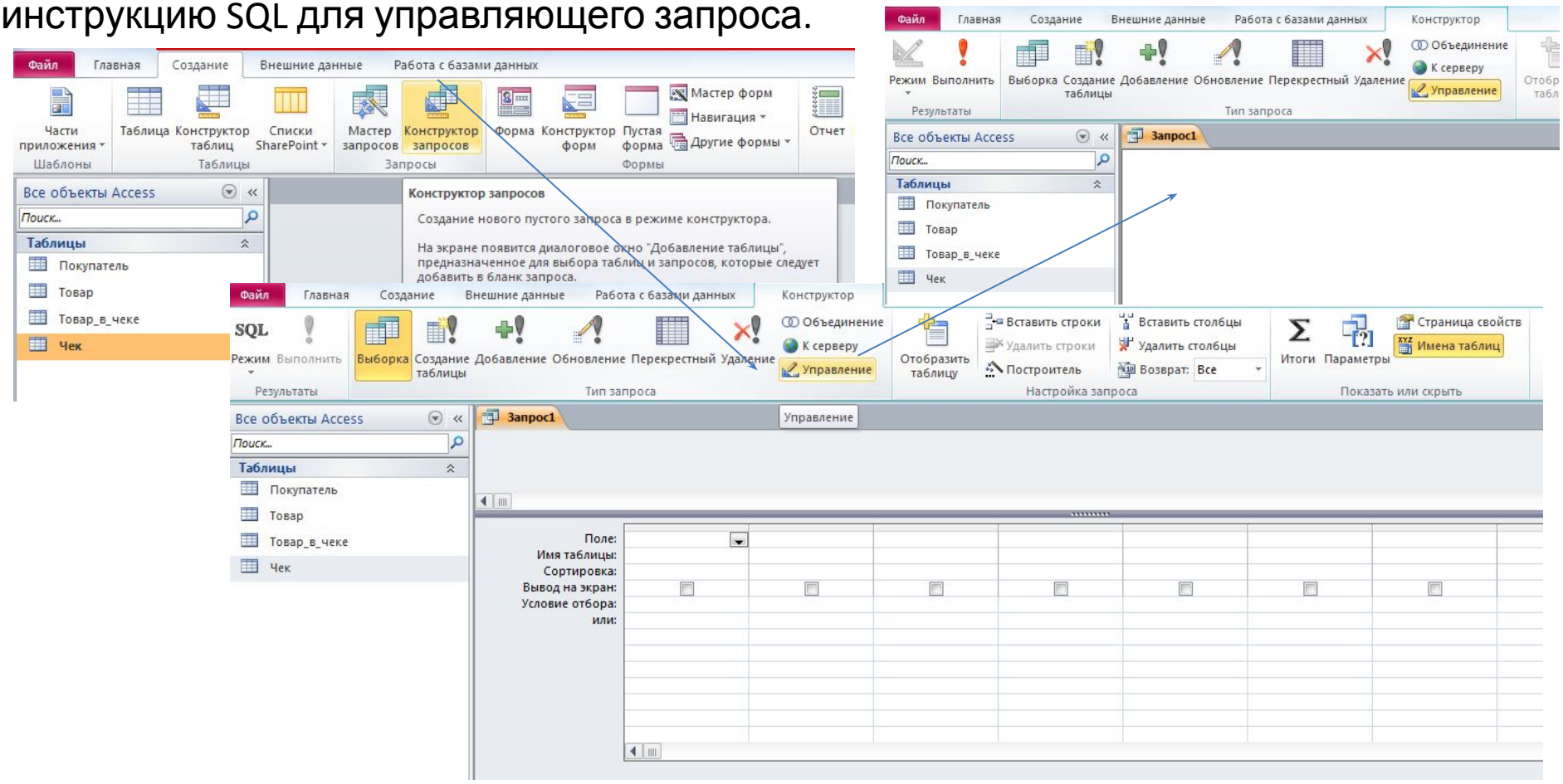
Демонстрация языка выполнена с использованием следующей модели



SQL в Access

Многие СУБД поддерживают интерактивный режим работы с интерфейсом в виде командной строки, в которой можно вводить команды языка SQL и выполнять их по нажатию Enter. Такого режима в Access нет, но возможность вводить и выполнять команды SQL имеется, например, в SQL-окне конструктора запросов. Команды SQL могут также быть частью программного кода на VBA.

На вкладке ***Создание*** выбираем ***Конструктор Запросов***. Закрываем окно ***Добавление таблицы*** и выбираем режим ***Управление***. Далее вводится инструкцию SQL для управляющего запроса.



SQL в Access

Каждый управляющий запрос может содержать только одну управляющую инструкцию.

При описании языка используются термины ***инструкция*** и ***предложение***. **Инструкция** – это полностью законченная команда языка SQL. Инструкции заканчиваются точкой с запятой.

Предложение – это часть инструкции, имеющая самостоятельное значение.

Работа с таблицами

Создание таблиц, индексов и других объектов можно осуществлять с помощью удобного конструктора, однако есть возможность использовать для этого управляющие запросы SQL. Управляющие запросы SQL используются для создания, удаления или изменения таблиц или для создания индексов в текущей базе данных.

Microsoft Access поддерживает следующие управляющие инструкции SQL:

- **CREATE TABLE** — создает таблицу;
- **ALTER TABLE** — добавляет новое поле или ограничение в существующую таблицу;
- **DROP TABLE** — удаляет таблицу из базы данных;
- **DROP INDEX** — удаляет индекс, определенный для поля или группы полей;
- **CREATE INDEX** — создает индекс для поля или группы полей.

Создание таблиц

Новая таблица создается инструкцией CREATE TABLE.

Синтаксис инструкции:

CREATE [TEMPORARY] TABLE таблица

(поле_1 тип [(размер)] [NOT NULL] [WITH COMPRESSION | WITH COMP] [индекс_1]

[, поле_2 тип [(размер)] [NOT NULL] [индекс_2] [, ...]]

[, CONSTRAINT составнойИндекс [, ...]]);

Ключевые слова при описании синтаксиса записываются заглавными буквами, необязательные элементы заключаются в квадратные скобки, вертикальная черта разделяет различные варианты написания

Элемент	Описание
таблица	Имя создаваемой таблицы.
поле_1, поле_2	Имена одного или нескольких полей, создаваемых в новой таблице. Таблица должна содержать хотя бы одно поле.
тип	Тип данных поля в новой таблице.
размер	Размер поля в знаках (только для текстовых и двоичных полей).
индекс_1, индекс_2	Предложение CONSTRAINT, предназначенное для создания простого индекса.
составнойИндекс	Предложение CONSTRAINT, предназначенное для создания составного индекса.

Создание таблиц

Инструкция **CREATE TABLE** используется для описания новой таблицы, ее полей и индексов. Если для поля добавлено ограничение **NOT NULL**, то при добавлении новых записей это поле должно содержать непустое значение.

Предложение **CONSTRAINT** устанавливает различные ограничения на поле и может быть использовано для определения ключа. Кроме того, для создания ключа или дополнительного индекса для существующей таблицы можно использовать инструкцию **CREATE INDEX**.

Допускается использование ограничения **NOT NULL** для одиночного поля, а также внутри именованного предложения **CONSTRAINT**, которое применяется к одиночному полю или к именованному предложению **CONSTRAINT**, предназначенному для создания составного индекса. Однако ограничение **NOT NULL** можно наложить на поле только один раз. При попытке применить это ограничение несколько раз возникает ошибка выполнения.

Создаваемая временная (**TEMPORARY**) таблица будет доступна только в том сеансе, где эта таблица была создана. По завершении данного сеанса она автоматически удаляется. Временные таблицы могут быть доступны для нескольких пользователей.

Использование атрибута **WITH COMPRESSION** допускается только для типов данных **CHARACTER** и **MEMO** (он же **TEXT**) и их синонимов.

Создание таблиц

Атрибут **WITH COMPRESSION** был добавлен к столбцам **CHARACTER** вследствие перехода к формату представления знаков Юникод. Каждый знак в формате Юникод всегда кодируется с помощью двух байтов. Для существующих баз данных Microsoft Jet, содержащих в основном символьные данные, это может означать увеличение размера файла базы данных примерно в два раза после преобразования в формат Microsoft Jet версии 4.0. Тем не менее, для многих наборов символов, ранее обозначавшихся как однобайтовые наборы символов (SBCS), представление в формате Unicode (SBCS) может быть без труда сжато до одного байта. Если столбец **CHARACTER** был определен с этим атрибутом, то при сохранении в нем данных осуществляется их автоматическое сжатие, а при извлечении данных – обратная операция.

Столбцы **MEMO** также могут быть определены для хранения данных в сжатом формате. Однако при этом действует одно ограничение. Сжатию подвергаются только те столбцы типа **MEMO**, которые в сжатом виде имеют размер не более 4096 байтов. Все остальные столбцы **MEMO** не сжимаются. Это означает, что для данной таблицы и данного столбца **MEMO** этой таблицы одни данные могут

Создание таблиц (упрощенная схема)

CREATE TABLE таблица
(столбец1 тип_данных1 [ограничение_столбца1],
столбец2 тип_данных2 [ограничение_столбца2], …
столбецN тип_данныхN [ограничение_столбцаN]
[, ограничение_таблицы1]
[, ограничение_таблицы2]
…
[, ограничение_таблицыN]);

Создание новой таблицы без ограничений

CREATE TABLE таблица
(столбец1 тип_данных1,
столбец2 тип_данных2, …
столбецN тип_данныхN);

Присвоение названия ограничению:

CONSTRAINT имя_ограничения

Ограничения

- NOT NULL - Не разрешает присваивать столбцу значение null
- DEFAULT - Задает для столбца значение по умолчанию
- PRIMARY KEY - Задает столбец (столбцы) первичного ключа для таблицы
- FOREIGN KEY - Задает столбец (столбцы) вторичного ключа для таблицы
- UNIQUE - Не разрешает добавлять в столбец повторяющиеся значения
- CHECK - Ограничивает значения, которые могут добавляться в столбец, с помощью логических выражений

- Ограничения бывают двух типов:
  - ***ограничение столбца*** является частью описания столбца и действует только для данного столбца;
  - ***ограничение таблицы*** не зависит от ограничений столбца и может влиять на несколько столбцов в таблице. Чтобы включить в ограничения требования для нескольких столбцов,

Ограничения NOT NULL и DEFAULT. Задание первичного ключа

- **Установка ограничения NOT NULL для столбца**

  [CONSTRAINT имя_ограничения]

  [NOT] NULL

- **Присвоение значения по умолчанию для столбца с помощью ограничения DEFAULT**

  [CONSTRAINT имя_ограничения]

  DEFAULT выражение

- **Задание первичного ключа с помощью ограничения PRIMARY KEY**
  - **В качестве ограничения столбца:**

  [CONSTRAINT имя_ограничения]

  PRIMARY KEY
  - **В качестве ограничения таблицы:**

  [CONSTRAINT имя_ограничения]

  PRIMARY KEY (ключевой_столбец)
  - **Задание сложного первичного ключа в качестве ограничения таблицы:**

  [CONSTRAINT имя_ограничения]

  PRIMARY KEY (ключевой_столбец1, ключевой_столбец2, ...)

Задание внешнего ключа с помощью ограничения FOREIGN KEY

- Создание простого внешнего ключа в качестве ограничения столбца

  [CONSTRAINT имя_ограничения]

  REFERENCES связанная_таблица (связанный_столбец)

- Создание простого внешнего ключа в качестве ограничения таблицы

  [CONSTRAINT имя_ограничения]

  FOREIGN KEY (ключевой_столбец)

  REFERENCES связанная_таблица (связанный_столбец)

- Задание сложного внешнего ключа в качестве ограничения таблицы

  [CONSTRAINT имя_ограничения]

  FOREIGN KEY (ключевые_столбцы)

  REFERENCES связанная_таблица (связанные_столбцы)

Изменение и удаление значений ключа

- При изменении или удалении значения ключа (в родительской таблице), на которое указывает значение внешнего ключа, для задания действия, в ограничении **FOREIGN KEY** указывается предложение:
  - **ON UPDATE** действие
  - ON DELETE действие
- Виды действий:
  - CASCADE заменит (удалит) значения внешних ключей в соответствии с новым (удаленным) значением первичного ключа;
  - SET NULL заменит значения внешних ключей на NULL;
  - SET DEFAULT заменит значения внешних ключей значениями по умолчанию;
  - NO ACTION выдаст ошибку для внешнего ключа. Эта установка задается по умолчанию.

Уникальные значения и ограничения на столбцы

- Присвоение уникальных значений с помощью ограничения UNIQUE
    - Создание простого ограничения уникальности в качестве ограничения столбца
  - [CONSTRAINT имя_ограничения] UNIQUE
    - Создание простого ограничения уникальности в качестве ограничения таблицы
  - [CONSTRAINT имя_ограничения]
  - UNIQUE (уникальный_столбец)
    - Задание сложного ограничения уникальности в качестве ограничения таблицы
  - [CONSTRAINT имя_ограничения]
  - UNIQUE (уникальные_столбцы)
- Проверка значений столбца с помощью ограничения CHECK
  - [CONSTRAINT имя_ограничения]
  - CHECK (условия)

Создание таблиц

Пример: создание таблица ***Друзья***.

CREATE TABLE Друзья

([Код] integer, [Фамилия] text, [Имя] text, [ДеньРождения] date,

[Телефон] text (10), [Примечания] memo,

CONSTRAINT [Индекс1] PRIMARY KEY ([Код]));

На рисунке показана созданная таблица в режиме конструктора. Свойству **Индексированное поле** поля *Код* установлено значение **Да (Совпадение не допускаются)**. Размер текстовых полей инструкцией SQL установлен максимально возможный.

Удаление таблиц

Пример: удаление таблицы *Друзья*.

DROP TABLE Друзья;

Создание индекса

При помощи индексов ускоряется сортировка и поиск записей. Индексы таблиц Microsoft Access используются так же, как предметные указатели в книгах: при поиске данных выполняется их поиск в индексе. Индексы можно создавать по одному или нескольким полям. Составные индексы позволяют пользователю различать записи, в которых первые поля могут иметь одинаковые значения. В основном требуется индексировать поля, в которых часто осуществляется поиск.

Однако индексы могут замедлить выполнение некоторых запросов на изменение, например, запросов на добавление, при выполнении которых требуется обновление индексов многих полей.

Поля первичного ключа таблиц индексируются автоматически, а поля с типом данных **Поле объекта OLE** индексировать нельзя. Для остальных полей индексирование используется, если выполняются следующие условия.

Если предполагается частое выполнение одновременной сортировки или поиска в нескольких полях, можно создать для этих полей составной индекс. Например, если в одном и том же запросе часто задаются условия для полей *Имя* и *Фамилия*, то для этих двух полей имеет смысл создать составной индекс.

При сортировке таблицы по составному индексу Microsoft Access сначала выполняет сортировку по первому полю, определенному для данного индекса. Если в первом поле содержатся записи с повторяющимися значениями, то выполняется сортировка по второму полю, определенному для данного индекса, и так далее.

Создание индекса

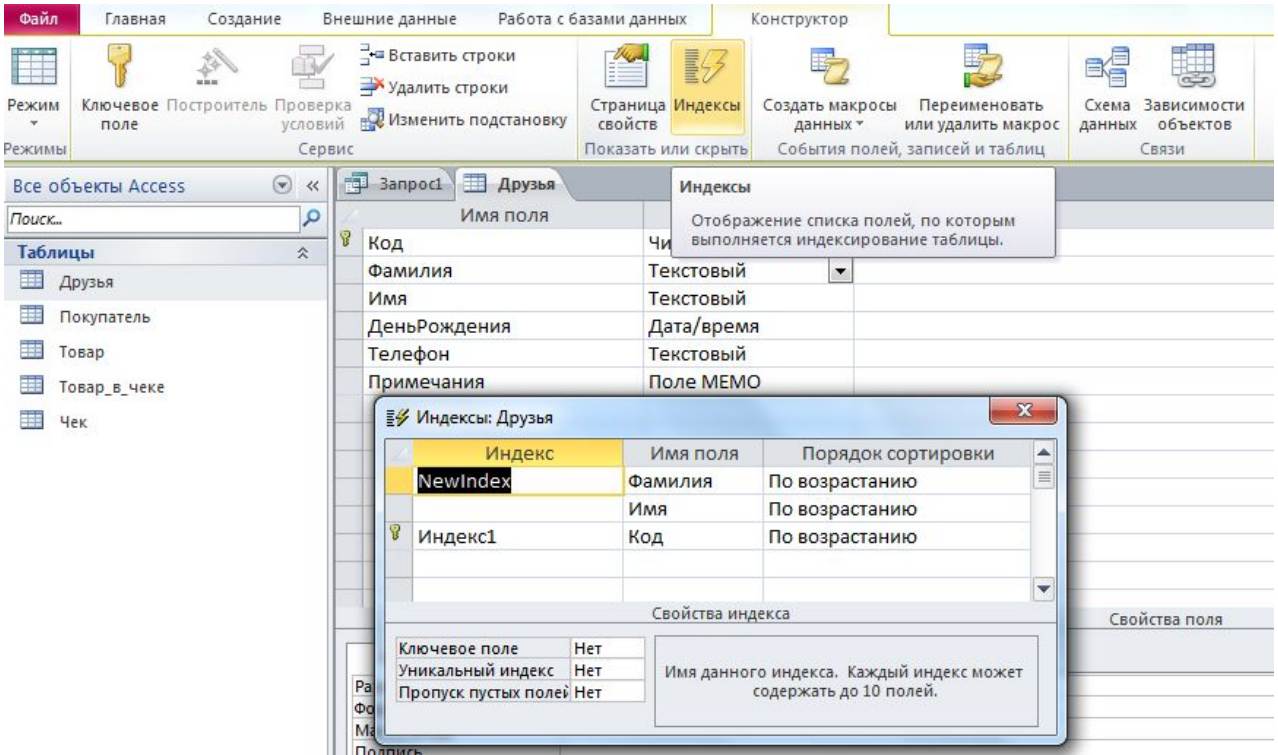
Пример: создание составного индекса по полям *Фамилия* и *Имя*.

```
CREATE INDEX NewIndex ON Друзья ([Фамилия], [Имя]);
```

Для просмотра созданных индексов откроем таблицу *Друзья* в режиме конструктора и нажмем кнопку 

. Откроется окно с перечнем индексов и полей, по которым производится индексация.

Для создания индекса нужно ввести его имя в столбце **Индекс**, а в столбце **Имя поля** выбрать из списка поле таблицы, по которому будет производиться индексация. Для удаления индекса нужно очистить содержимое строк, содержащих описание индекса.



Создание таблиц на основе существующей.

Создание новой таблицы на основе существующей с помощью команды SELECT INTO

```
SELECT столбцы
INTO новая_таблица
FROM существующая_таблица
[WHERE условия_поиска];
```

Константы и NULL-значение

- Числовые значения
  - 0 1 2 3 4 5 6 7 8 9 + - $. E e
- Булевы значения, строковые константы и даты
  - TRUE ‘Hello’ 12-04-2011 14:40 ‘Д’’Артаньян’
- NULL – пустое или несуществующее значение
  - Значения типа NULL нельзя помещать в столбцы, определенные как NOT NULL
  - Значения типа NULL не равны друг другу
  - Игнорируется при вычислении агрегатных значений
  - При группировке все найденные значения NULL рассматриваются как одна группа

Добавление строк

Добавление строки с помощью положения столбца

```
INSERT INTO  таблица
VALUES(значение1,  значение2, …,  значениеN);
```

Пример: добавление нового покупателя

```
INSERT INTO  Покупатель
VALUES(11111,'Иванов','Николай','Петрович',1234565,'Муж','01-01-2001')
;
```

Добавление строки с помощью названий столбцов

```
INSERT INTO  таблица
 (Штрих-код,  Номер_чека, Скидка)
  VALUES(1,  1, 0);
```

Пример: добавление записи в таблицу Товар_в_чеке

```
INSERT  INTO Товар_в_чеке
 (Штрих_код,  Номер_чека, Скидка)
  VALUES(1,  1, 0);
```

Замечание: в записях названия «тире» не воспринимается, то есть вместо

Операторы

- Оператор – это символ или имя, обозначающий действие, выполняемое над одним или несколькими выражениями.
- Арифметические операторы

  + - * / (+ и – можно применять к датам)
- Оператор конкатенации (&) соединяет две отдельных текстовых строки в одно строковое значение.
- Оператор присваивания (=)
- Унарные операторы

  + числовое значение становится положительным

  - числовое значение становится отрицательным

Операторы

- Операторы сравнения

  TRUE, FALSE, UNKNOWN, NULL

Оператор	Описание
=	Равно
>	Больше
<	Меньше
>=	Больше или равно
<=	Меньше или равно
<>	Не равно
IS NULL	Имеет ли значение NULL
IS NOT NULL	Не имеет ли значение NULL

Операторы

Логические операторы

Оператор	Описание
ALL	TRUE, если весь набор сравнений даст результат TRUE
AND	TRUE, если оба булевых выражения дают результат TRUE
ANY	TRUE, если хотя бы одно сравнение из набора даст результат TRUE
BETWEEN	TRUE, если операнд находится внутри диапазона
EXISTS	TRUE, если подзапрос возвращает хотя бы одну строку
IN	TRUE, если операнд равен одному выражению из списка или одной или нескольким строкам, возвращаемым подзапросом
LIKE	TRUE, если операнд совпадает с шаблоном
NOT	Обращает значение любого другого булевого оператора
OR	TRUE, если любое булево выражение равно TRUE
SOME	TRUE, если несколько сравнений из набора дают результат TRUE

Приоритет операторов

1. () (выражения, стоящие в скобках)
2. +, - (унарные операторы)
3. *, / (математические операторы)
4. +, - (арифметические операторы)
5. =, >, <, >=, <=, <> (операторы сравнения)
6. IS NULL
7. NOT
8. AND
9. ALL, ANY, BETWEEN, IN, LIKE, OR, SOME
0. = (присваивание значения переменной)

Изменение и удаление строк

Изменение строк с помощью команды UPDATE

```
UPDATE  таблица
SET  столбец  =  новое_значение
[WHERE  условия_выбора_строк];
```

Пример: запрос, меняющий фамилии

```
UPDATE  Покупатель
SET  Фамилия  = 'Иванченко'
WHERE  Фамилия='Иванов';
```

Удаление строк с помощью команды DELETE

```
DELETE FROM  таблица
[WHERE  условия_выбора_строк];
```

Пример: удалить все записи заданного чека

```
DELETE FROM  Товар_в_чеке
WHERE  Номер_чека=1;
```

Инструкция SELECT

По этой инструкции ядро базы данных Microsoft Jet возвращает данные из базы данных в виде набора записей.

Синтаксис структуры оператора SELECT имеет вид:

SELECT список столбцов, которые включаются в результат выборки

{INTO новая таблица**}**

FROM список таблиц, из которых производится выборка

[WHERE условия выборки**]**

[GROUP BY список столбцов, по которым происходит группировка**]**

[HAVING условия группировки**]**

[ORDER BY выражение, в соответствии с которым выполняется сортировка возвращаемых данных**];**

В квадратные скобки заключены необязательные выражения оператора **SELECT**.

Если не указано имя таблицы для вывода, то результат формируется во временную таблицу.

При выполнении этой операции ядро базы данных находит указанную таблицу или таблицы, извлекает заданные столбцы, выделяет строки, соответствующие условию отбора, и сортирует или группирует результирующие строки в указанном порядке. Инструкции **SELECT** не изменяют данные в базе данных.

Обычно слово **SELECT** является первым словом инструкции **SQL**. Большая

Отбор данных из одной таблицы. Простая выборка

Вывод нескольких столбцов таблицы (проекция).

SELECT столбец 1, …, столбец N
FROM таблица;

Замечание: порядок столбцов может быть произвольный.

Пример: получение ФИО всех покупателей и их даты рождения.

SELECT Имя, Отчество, Фамилия,Год_рождения
FROM Покупатель;

Номер_кар	Фамилия	Имя	Отчество	Телефон	Пол	Год_рождения
11111	Иванченко	Николай	Петрович	1234565	Муж	01.01.2001
64363	Иванченко	Иван	Сидорович	747474	Муж	03.08.1977
111111	Иванченко	Николай	Петрович	1234565	Муж	08.07.1894
1111111	Иванченко	Николай	Николаевич	1234565	Муж	01.01.2001
1133411	Анченко	Николай	Петрович	1234565	Муж	01.01.2001
3636336	Иванов	Степан	Николаевич	5735734	Муж	11.11.2011
5346346	Иванова	Марина	Олеговна	75745745	Жен	12.12.2012
7464798	Сидоров	Иван	Петрович	3456789	Муж	01.02.2013
7576447	Иванченко	Николай	Петрович	1234565	Муж	01.05.2001
23456789	Петров	Петр	Петрович	87435454	Муж	22.12.2012
65467347	Иванченко	Александра	Николаевна	75858574	Жен	11.11.2011
123456789	Иванченко	Иван	Иванович	1234567	Муж	08.12.1934

Имя	Отчество	Фамилия	Год_рождения
Николай	Петрович	Иванченко	01.01.2001
Иван	Сидорович	Иванченко	03.08.1977
Николай	Петрович	Иванченко	08.07.1894
Николай	Николаевич	Иванченко	01.01.2001
Николай	Петрович	Анченко	01.01.2001
Степан	Николаевич	Иванов	11.11.2011
Марина	Олеговна	Иванова	12.12.2012
Иван	Петрович	Сидоров	01.02.2013
Николай	Петрович	Иванченко	01.05.2001
Петр	Петрович	Петров	22.12.2012
Александра	Николаевна	Иванченко	11.11.2011
Иван	Иванович	Иванченко	08.12.1934

Отбор данных из одной таблицы. Простая выборка

Вывод всей таблицы

```
SELECT *
FROM таблица;
```

Замечание: * обозначает все имена столбцов таблицы, при этом порядок вывода столбцов соответствует порядку, в котором они определялись при создании таблицы.

Пример: Получение полной информации о покупателях.

```
SELECT *
FROM Покупатель;
```

Номер_кар	Фамилия	Имя	Отчество	Телефон	Пол	Год_рождения
11111	Иванченко	Николай	Петрович	1234565	Муж	01.01.2001
111111	Иванченко	Николай	Петрович	1234565	Муж	08.07.1894
1111111	Иванченко	Николай	Николаевич	1234565	Муж	01.01.2001
7464798	Сидоров	Иван	Петрович	3456789	Муж	01.02.2013
23456789	Петров	Петр	Петрович	87435454	Муж	22.12.2012
65467347	Иванова	Александра	Николаевна	75858574	Жен	11.11.2011
123456789	Иванченко	Иван	Иванович	1234567	Муж	08.12.1934

Отбор данных из одной таблицы. Простая выборка

Исключение дублирующей информации

SELECT DISTINCT столбец1,… , столбец N

FROM Покупатель;

Замечание: DISTINCT исключает дублирование кортежей при проекции столбцов, в случае его отсутствия кортежи могут дублироваться.

Пример: получить разные фамилии клиентов.

SELECT DISTINCT Фамилия

FROM Покупатель;

Фамилия
Анченко
Иванов
Иванова
Иванченко
Петров
Сидоров

SELECT Фамилия

FROM Покупатель;

Фамилия
Иванченко
Иванченко
Иванченко
Иванченко
Анченко
Иванов
Иванова
Сидоров
Иванченко
Петров
Иванченко
Иванченко

Номер_кар	Фамилия	Имя	Отчество	Телефон	Пол	Год_рождения
11111	Иванченко	Николай	Петрович	1234565	Муж	01.01.2001
64363	Иванченко	Иван	Сидорович	747474	Муж	03.08.1977
111111	Иванченко	Николай	Петрович	1234565	Муж	08.07.1894
1111111	Иванченко	Николай	Николаевич	1234565	Муж	01.01.2001
1133411	Анченко	Николай	Петрович	1234565	Муж	01.01.2001
3636336	Иванов	Степан	Николаевич	5735734	Муж	11.11.2011
5346346	Иванова	Марина	Олеговна	75745745	Жен	12.12.2012
7464798	Сидоров	Иван	Петрович	3456789	Муж	01.02.2013
7576447	Иванченко	Николай	Петрович	1234565	Муж	01.05.2001
23456789	Петров	Петр	Петрович	87435454	Муж	22.12.2012
65467347	Иванченко	Александра	Николаевна	75858574	Жен	11.11.2011
123456789	Иванченко	Иван	Иванович	1234567	Муж	08.12.1934

Запросы с вычисляемыми полями

Пример: узнать суммарную стоимость каждой единицы товара.

```
SELECT Название, 4 AS Количество, Цена, "=" AS Равно, Количество*Цена AS Сумма
FROM Товар;
```

Замечания:

)Конструкция типа «Значение переменной AS Имя столбца» означает, что создается столбец (Имя столбца), в каждой ячейке которого лежит значение (Значение переменной).

)«Количество*Цена AS Сумма» означает, что в столбец Сумма будет помещено значение, являющееся результатом вычисления Количество*Цена для соответствующей строки.

Штрих-код	Название	Цена	Наличие_ак
1	хлеб	23,00р.	☐
2	сок	55,00р.	☑
3	картофель	20,00р.	☐
4	хлеб	30,00р.	☐
5	сок	70,00р.	☐
6	хлеб	22,00р.	☑
7	сок	58,00р.	☑

Название	Количество	Цена	Равно	Сумма
хлеб	4	23,00р.	=	92,00р.
сок	4	55,00р.	=	220,00р.
картофель	4	20,00р.	=	80,00р.
хлеб	4	30,00р.	=	120,00р.
сок	4	70,00р.	=	280,00р.
хлеб	4	22,00р.	=	88,00р.
сок	4	58,00р.	=	232,00р.

Запросы с соединением данных

SELECT Название &': сумма за 4 единицы равна ' & 4*Цена AS Сумма
FROM Товар;

Штрих-код	Название	Цена	Наличие_ан
1	хлеб	23,00р.	☐
2	сок	55,00р.	☑
3	картофель	20,00р.	☐
4	хлеб	30,00р.	☐
5	сок	70,00р.	☐
6	хлеб	22,00р.	☑
7	сок	58,00р.	☑

Сумма
хлеб: сумма за 4 единицы равна 92
сок: сумма за 4 единицы равна 220
картофель: сумма за 4 единицы равна 80
хлеб: сумма за 4 единицы равна 120
сок: сумма за 4 единицы равна 280
хлеб: сумма за 4 единицы равна 88
сок: сумма за 4 единицы равна 232

Запросы с условием отбора

SELECT столбец 1, …, столбец N
FROM таблица
WHERE условие отбора;

WHERE условие_поиска [{AND | OR | NOT} условие_поиска […],

условие_поиска:

значение { = | <> | < | <= | > | >= } { значение | (подзапрос) }

значение_1 [NOT] BETWEEN значение_2 AND значение_3

значение [NOT] IN { (константа [, константа]…) | (подзапрос) }

значение IS [NOT] NULL

[таблица.] столбец [NOT] LIKE ‘строка_символов’ [ESCAPE ‘символ’]

EXISTS (подзапрос)

Критерий отбора строк формируется из одного или нескольких условий, соединенных логическими операторами:

AND – выполняются оба условия

OR - выполняется одно из условий

AND NOT – выполняется первое, но не выполняется второе

OR NOT – или выполняется первое или не выполняется второе

Запросы с условием отбора

Пример: найти всех покупателей, у которых фамилия Иванченко и которых зовут Иван.

```
SELECT *
FROM Покупатель
WHERE Фамилия='Иванченко' AND Имя='Иван';
```

Номер_кар	Фамилия	Имя	Отчество	Телефон	Пол	Год_рождения
11111	Иванченко	Николай	Петрович	1234565	Муж	01.01.2001
64363	Иванченко	Иван	Сидорович	747474	Муж	03.08.1977
111111	Иванченко	Николай	Петрович	1234565	Муж	08.07.1894
1111111	Иванченко	Николай	Николаевич	1234565	Муж	01.01.2001
1133411	Анченко	Николай	Петрович	1234565	Муж	01.01.2001
3636336	Иванов	Степан	Николаевич	5735734	Муж	11.11.2011
5346346	Иванова	Марина	Олеговна	75745745	Жен	12.12.2012
7464798	Сидоров	Иван	Петрович	3456789	Муж	01.02.2013
7576447	Иванченко	Николай	Петрович	1234565	Муж	01.05.2001
23456789	Петров	Петр	Петрович	87435454	Муж	22.12.2012
65467347	Иванченко	Александра	Николаевна	75858574	Жен	11.11.2011
123456789	Иванченко	Иван	Иванович	1234567	Муж	08.12.1934

Номер_кар	Фамилия	Имя	Отчество	Телефон	Пол	Год_рождения
123456789	Иванченко	Иван	Иванович	1234567	Муж	08.12.1934
64363	Иванченко	Иван	Сидорович	747474	Муж	03.08.1977

Запросы с условием отбора

Пример: найти всех покупателей, у которых фамилия Иванченко или которых не зовут Иван.

```
SELECT *
FROM Покупатель
WHERE Фамилия='Иванченко' OR NOT Имя='Иван';
```

Номер_кар	Фамилия	Имя	Отчество	Телефон	Пол	Год_рождения
11111	Иванченко	Николай	Петрович	1234565	Муж	01.01.2001
64363	Иванченко	Иван	Сидорович	747474	Муж	03.08.1977
111111	Иванченко	Николай	Петрович	1234565	Муж	08.07.1894
1111111	Иванченко	Николай	Николаевич	1234565	Муж	01.01.2001
1133411	Анченко	Николай	Петрович	1234565	Муж	01.01.2001
3636336	Иванов	Степан	Николаевич	5735734	Муж	11.11.2011
5346346	Иванова	Марина	Олеговна	75745745	Жен	12.12.2012
7464798	Сидоров	Иван	Петрович	3456789	Муж	01.02.2013
7576447	Иванченко	Николай	Петрович	1234565	Муж	01.05.2001
23456789	Петров	Петр	Петрович	87435454	Муж	22.12.2012
65467347	Иванченко	Александра	Николаевна	75858574	Жен	11.11.2011
123456789	Иванченко	Иван	Иванович	1234567	Муж	08.12.1934

Номер_кар	Фамилия	Имя	Отчество	Телефон	Пол	Год_рождения
7576447	Иванченко	Николай	Петрович	1234565	Муж	01.05.2001
3636336	Иванов	Степан	Николаевич	5735734	Муж	11.11.2011
5346346	Иванова	Марина	Олеговна	75745745	Жен	12.12.2012
1133411	Анченко	Николай	Петрович	1234565	Муж	01.01.2001
123456789	Иванченко	Иван	Иванович	1234567	Муж	08.12.1934
23456789	Петров	Петр	Петрович	87435454	Муж	22.12.2012
65467347	Иванченко	Александра	Николаевна	75858574	Жен	11.11.2011
1111111	Иванченко	Николай	Николаевич	1234565	Муж	01.01.2001
111111	Иванченко	Николай	Петрович	1234565	Муж	08.07.1894
11111	Иванченко	Николай	Петрович	1234565	Муж	01.01.2001
64363	Иванченко	Иван	Сидорович	747474	Муж	03.08.1977

Запросы с условием отбора

Оператор BETWEEN…AND…

Нахождение значений в интервале от и до.

Пример: вывести товары, ценовая категория которых от 22 до 56 рублей.

```
SELECT *
FROM Товар
WHERE Цена BETWEEN 22 AND 56;
```

Штрих-код	Название	Цена	Наличие_ак
1	хлеб	23,00р.	☐
2	сок	55,00р.	☑
3	картофель	20,00р.	☐
4	хлеб	30,00р.	☐
5	сок	70,00р.	☐
6	хлеб	22,00р.	☑
7	сок	58,00р.	☑

Штрих-код	Название	Цена	Наличие_ак
1	хлеб	23,00р.	☐
2	сок	55,00р.	☑
4	хлеб	30,00р.	☐
6	хлеб	22,00р.	☑

Запросы с условием отбора

Оператор BETWEEN…AND…

Пример: вывести покупателей в заданные годы рождения.

```
SELECT *
FROM Покупатель
WHERE Год_рождения BETWEEN #01/01/2000# AND #01/01/2012#;
```

Номер_кар	Фамилия	Имя	Отчество	Телефон	Пол	Год_рождения
11111	Иванченко	Николай	Петрович	1234565	Муж	01.01.2001
64363	Иванченко	Иван	Сидорович	747474	Муж	03.08.1977
111111	Иванченко	Николай	Петрович	1234565	Муж	08.07.1894
1111111	Иванченко	Николай	Николаевич	1234565	Муж	01.01.2001
1133411	Анченко	Николай	Петрович	1234565	Муж	01.01.2001
3636336	Иванов	Степан	Николаевич	5735734	Муж	11.11.2011
5346346	Иванова	Марина	Олеговна	75745745	Жен	12.12.2012
7464798	Сидоров	Иван	Петрович	3456789	Муж	01.02.2013
7576447	Иванченко	Николай	Петрович	1234565	Муж	01.05.2001
23456789	Петров	Петр	Петрович	87435454	Муж	22.12.2012
65467347	Иванченко	Александра	Николаевна	75858574	Жен	11.11.2011
123456789	Иванченко	Иван	Иванович	1234567	Муж	08.12.1934

Номер_кар	Фамилия	Имя	Отчество	Телефон	Пол	Год_рождения
7576447	Иванченко	Николай	Петрович	1234565	Муж	01.05.2001
3636336	Иванов	Степан	Николаевич	5735734	Муж	11.11.2011
1133411	Анченко	Николай	Петрович	1234565	Муж	01.01.2001
65467347	Иванченко	Александра	Николаевна	75858574	Жен	11.11.2011
1111111	Иванченко	Николай	Николаевич	1234565	Муж	01.01.2001
11111	Иванченко	Николай	Петрович	1234565	Муж	01.01.2001

Запросы с условием отбора

Оператор IN

Определяет набор значений с помощью соответствующего набора данных, заключенные в круглые скобки и отделенные запятой.

Пример: найти покупателей с заданными фамилиями.

```
SELECT *
FROM Покупатель
WHERE Фамилия IN('Иванов','Иванова','Иванченко');
```

Номер_кар	Фамилия	Имя	Отчество	Телефон	Пол	Год_рождения
11111	Иванченко	Николай	Петрович	1234565	Муж	01.01.2001
64363	Иванченко	Иван	Сидорович	747474	Муж	03.08.1977
111111	Иванченко	Николай	Петрович	1234565	Муж	08.07.1894
1111111	Иванченко	Николай	Николаевич	1234565	Муж	01.01.2001
1133411	Анченко	Николай	Петрович	1234565	Муж	01.01.2001
3636336	Иванов	Степан	Николаевич	5735734	Муж	11.11.2011
5346346	Иванова	Марина	Олеговна	75745745	Жен	12.12.2012
7464798	Сидоров	Иван	Петрович	3456789	Муж	01.02.2013
7576447	Иванченко	Николай	Петрович	1234565	Муж	01.05.2001
23456789	Петров	Петр	Петрович	87435454	Муж	22.12.2012
65467347	Иванченко	Александра	Николаевна	75858574	Жен	11.11.2011
123456789	Иванченко	Иван	Иванович	1234567	Муж	08.12.1934

Номер_кар	Фамилия	Имя	Отчество	Телефон	Пол	Год_рождения
7576447	Иванченко	Николай	Петрович	1234565	Муж	01.05.2001
3636336	Иванов	Степан	Николаевич	5735734	Муж	11.11.2011
5346346	Иванова	Марина	Олеговна	75745745	Жен	12.12.2012
123456789	Иванченко	Иван	Иванович	1234567	Муж	08.12.1934
65467347	Иванченко	Александра	Николаевна	75858574	Жен	11.11.2011
1111111	Иванченко	Николай	Николаевич	1234565	Муж	01.01.2001
111111	Иванченко	Николай	Петрович	1234565	Муж	08.07.1894
11111	Иванченко	Николай	Петрович	1234565	Муж	01.01.2001
64363	Иванченко	Иван	Сидорович	747474	Муж	03.08.1977

Запросы с условием отбора

Оператор LIKE

Применяется только к текстовым данным, используется для нахождения подстрок, совпадающих с заданным образцом.

Замечание: символ ‘?’ заменяет любой одиночный символ, знак ’*’ заменяет последовательность произвольного числа символов (в том числе и их отсутствие).

Пример: найти покупателей, н

```
SELECT *
FROM Покупатель
WHERE Фамилия LIKE 'Иван*';
```

Номер_кар	Фамилия	Имя	Отчество	Телефон	Пол	Год_рождения
11111	Иванченко	Николай	Петрович	1234565	Муж	01.01.2001
64363	Иванченко	Иван	Сидорович	747474	Муж	03.08.1977
111111	Иванченко	Николай	Петрович	1234565	Муж	08.07.1894
1111111	Иванченко	Николай	Николаевич	1234565	Муж	01.01.2001
1133411	Анченко	Николай	Петрович	1234565	Муж	01.01.2001
3636336	Иванов	Степан	Николаевич	5735734	Муж	11.11.2011
5346346	Иванова	Марина	Олеговна	75745745	Жен	12.12.2012
7464798	Сидоров	Иван	Петрович	3456789	Муж	01.02.2013
7576447	Иванченко	Николай	Петрович	1234565	Муж	01.05.2001
23456789	Петров	Петр	Петрович	87435454	Муж	22.12.2012
65467347	Иванченко	Александра	Николаевна	75858574	Жен	11.11.2011
123456789	Иванченко	Иван	Иванович	1234567	Муж	08.12.1934

Номер_кар	Фамилия	Имя	Отчество	Телефон	Пол	Год_рождения
7576447	Иванченко	Николай	Петрович	1234565	Муж	01.05.2001
3636336	Иванов	Степан	Николаевич	5735734	Муж	11.11.2011
5346346	Иванова	Марина	Олеговна	75745745	Жен	12.12.2012
123456789	Иванченко	Иван	Иванович	1234567	Муж	08.12.1934
65467347	Иванченко	Александра	Николаевна	75858574	Жен	11.11.2011
1111111	Иванченко	Николай	Николаевич	1234565	Муж	01.01.2001
111111	Иванченко	Николай	Петрович	1234565	Муж	08.07.1894
11111	Иванченко	Николай	Петрович	1234565	Муж	01.01.2001
64363	Иванченко	Иван	Сидорович	747474	Муж	03.08.1977

Запросы с условием отбора

Оператор LIKE

Пример: найти покупателей, у которых первая буква неизвестна, потом идет ‘?ванов…’, в дальнейшем идут символы, или они могут отсутствовать.

```
SELECT *
FROM Покупатель
WHERE Фамилия LIKE '?ванов*';
```

Номер_кар	Фамилия	Имя	Отчество	Телефон	Пол	Год_рождения
11111	Иванченко	Николай	Петрович	1234565	Муж	01.01.2001
64363	Иванченко	Иван	Сидорович	747474	Муж	03.08.1977
111111	Иванченко	Николай	Петрович	1234565	Муж	08.07.1894
1111111	Иванченко	Николай	Николаевич	1234565	Муж	01.01.2001
1133411	Анченко	Николай	Петрович	1234565	Муж	01.01.2001
3636336	Иванов	Степан	Николаевич	5735734	Муж	11.11.2011
5346346	Иванова	Марина	Олеговна	75745745	Жен	12.12.2012
7464798	Сидоров	Иван	Петрович	3456789	Муж	01.02.2013
7576447	Иванченко	Николай	Петрович	1234565	Муж	01.05.2001
23456789	Петров	Петр	Петрович	87435454	Муж	22.12.2012
65467347	Иванченко	Александра	Николаевна	75858574	Жен	11.11.2011
123456789	Иванченко	Иван	Иванович	1234567	Муж	08.12.1934

Номер_кар	Фамилия	Имя	Отчество	Телефон	Пол	Год_рождения
3636336	Иванов	Степан	Николаевич	5735734	Муж	11.11.2011
5346346	Иванова	Марина	Олеговна	75745745	Жен	12.12.2012

Обобщение данных с помощью агрегатных функций

SUM – сумма

AVG – среднее значение

MAX – максимальное значение

MIN – минимальное значение

COUNT – количество

```
SELECT агрегатная_функция (название столбца)
FROM таблица;
```

Обобщение данных с помощью агрегатных функций

Штрих-код	Название	Цена	Наличие_ак
1	хлеб	23,00р.	☐
2	сок	55,00р.	☑
3	кортофель	20,00р.	☐
			☐

SELECT SUM(Цена) AS Сумма,AVG(Цена) AS Среднее, MAX (Цена) AS Максимум, MIN(Цена) AS Минимум, COUNT (Цена) AS Количество
FROM Товар;

Сумма	Среднее	Максимум	Минимум	Количество
98,00р.	32,67р.	55,00р.	20,00р.	3

Оператор GROUP BY

Производит вывод таблицы по группам, каждая из которых имеет одинаковые значения в столбце (столбцах), указанных в GROUP BY.

SELECT столбцы-критерии объединения в группу

FROM Таблица

GROUP BY столбцы, по которым идет объединение;

Пример: получение минимальной цены по всем товарам.

SELECT Название, MIN (Цена) AS МинЦена

FROM Товар

GROUP BY Название;

Штрих-код	Название	Цена	Наличие_ак
1	хлеб	23,00р.	☐
2	сок	55,00р.	☑
3	картофель	20,00р.	☐
4	хлеб	30,00р.	☐
5	сок	70,00р.	☐
6	хлеб	22,00р.	☑
7	сок	58,00р.	☑

Название	МинЦена
картофель	20,00р.
сок	55,00р.
хлеб	22,00р.

Оператор GROUP BY

Пример: получение минимальной цены по всем товарам с учетом наличия акции.

```
SELECT Название, MIN (Цена) AS МинЦена, Наличие_акции
FROM Товар
GROUP BY Название, Наличие_акции;
```

Штрих-код	Название	Цена	Наличие_ак
1	хлеб	23,00р.	☐
2	сок	55,00р.	☑
3	картофель	20,00р.	☐
4	хлеб	30,00р.	☐
5	сок	70,00р.	☐
6	хлеб	22,00р.	☑
7	сок	58,00р.	☑

Название	МинЦена	Наличие_ак
картофель	20,00р.	☐
сок	55,00р.	☑
сок	70,00р.	☐
хлеб	22,00р.	☑
хлеб	23,00р.	☐

Оператор GROUP BY с условием HAVING

HAVING – условия группировки (аналог WHERE в случае группировки).

SELECT столбцы-критерии объединения в группу
FROM Таблица
GROUP BY столбцы, по которым идет объединение
HAVING наложенные условия;

Пример: вывести минимальную цену товаров, которые продаются по акции.

```
SELECT Название, MIN (Цена) AS МинЦена
    FROM Товар
GROUP BY Название, Наличие_акции
HAVING Наличие_акции=TRUE;
```

Штрих-код	Название	Цена	Наличие_ак
1	хлеб	23,00р.	☐
2	сок	55,00р.	☑
3	картофель	20,00р.	☐
4	хлеб	30,00р.	☐
5	сок	70,00р.	☐
6	хлеб	22,00р.	☑
7	сок	58,00р.	☑

Название	МинЦена
сок	55,00р.
хлеб	22,00р.

Упорядочивание вывода полей. Оператор ORDER BY

SELECT столбцы-критерии объединения в группу

FROM Таблица

ORDER BY столбец1 критерий_упорядочивания,…, столбецN критерий_упорядочивания;

Критерий упорядочивания:

ASC – по возрастанию (по умолчанию)

DESC – по убыванию

Пример: получение минимальной цены по всем товарам с учетом наличия акции так, чтобы товары располагалась по убыванию, а цена – по возрастанию.

```
SELECT Название, MIN (Цена) AS МинЦена, Наличие_акции
FROM Товар
GROUP BY Название, Наличие_акции
ORDER BY Название, MIN (Цена) DESC;
```

Штрих-код	Название	Цена	Наличие_ак
1	хлеб	23,00р.	☐
2	сок	55,00р.	☑
3	картофель	20,00р.	☐
4	хлеб	30,00р.	☐
5	сок	70,00р.	☐
6	хлеб	22,00р.	☑
7	сок	58,00р.	☑

Название	МинЦена	Наличие_ак
картофель	20,00р.	☐
сок	70,00р.	☐
сок	55,00р.	☑
хлеб	23,00р.	☐
хлеб	22,00р.	☑

Особенности работы с несколькими таблицами

Использование уточненных имен:

Таблица.Столбец

Использование объединений – предложение **JOIN**

1) Таблицы всегда объединяются построчно при выполнении всевозможных условий, определенных в вашем запросе.

2) Строки, несоответствующие заданным условиям, могут быть как включены в объединение, так и исключены из него, в зависимости от типа этого объединения.

Объединением по равенству называется такое объединение, в условии которого применяется оператор равенства (=), чтобы группировать строки, имеющие равные значения в определенных столбцах, называемых связанными.

Тэта-объединение – более общий случай когда значения связанных столбцов сравнивают с применением любого оператора сравнения.

Связанные столбцы любого объединения чаще всего оказываются связанными ключевыми столбцами, но можно связывать любые столбцы, если их типы данных совместимы.

Создание объединений с помощью синтаксиса JOIN

SELECT столбцы
FROM таблица1 тип_JOIN таблица2
ON условие_объединения
[WHERE условие_поиска]
[GROUP BY условия_группировки]
[HAVING условия_поиска]
[ORDER BY столбец_сортировки];

Типы объединений JOIN:

1) **CROSS JOIN** – для перекрестного объединения;
2) **NATURAL JOIN** – для естественного объединения;
3) **INNER JOIN – для внутреннего объединения;**
4) **LEFT [OUTER] JOIN – для левого внешнего объединения;**
5) **RIGHT [OUTER] JOIN – для правого внешнего объединения;**
6) **FULL [OUTER] JOIN** – для полного внешнего объединения.

Последовательность выполнения запроса

Когда СУБД обрабатывает объединения, она подчиняется определенной последовательности шагов, которая не только относится к обработке объединений, но и определяет алгоритм выполнения всего запроса:

1)Применить условия объединения, заданные предложением JOIN.

2) Применить условия объединения и поиска, заданные предложением WHERE.

3) Сгруппировать строки в соответствии с предложением GROUP BY.

4) Применить к группам условия поиска, заданные предложением HAVING.

5) Отсортировать результат в соответствии с предложением ORDER BY.

Декартово произведение на языке SQL

CROSS JOIN – перекрестное соединение, которое просто формирует декартово произведение таблиц (это все возможные комбинации строк двух таблиц так, что каждая строка первой таблицы объединяется с каждой строкой второй таблицы).

```
SELECT Покупатель.*,Товар.*
FROM Покупатель,Товар;
```

ил
и

```
SELECT Покупатель.*,Товар.*
FROM Покупатель CROSS JOIN
Товар;
```

Замечания:

)В случае запроса с несколькими таблицами указание столбцов таблицы имеет вид: **Название_таблицы.Название_стобца .**

)Синтаксис **Название_таблицы.*** означает использование всех столбцов в таблице.

Номер_кар	Фамилия	Имя	Отчество	Телефон	Пол	Год_рождения
11111	Иванченко	Николай	Петрович	1234565	Муж	01.01.2001
7464798	Сидоров	Иван	Петрович	3456789	Муж	01.02.2013
23456789	Петров	Петр	Петрович	87435454	Муж	22.12.2012
65467347	Иванова	Александра	Николаевна	75858574	Жен	11.11.2011

про

Штрих-код	Название	Цена	Наличие_ак
1	хлеб	23,00р.	☐
2	сок	55,00р.	☑
3	картофель	20,00р.	☐

Номер_кар	Фамилия	Имя	Отчество	Телефон	Пол	Год_рождения	Штрих-код	Название	Цена	Наличие_ак
23456789	Петров	Петр	Петрович	87435454	Муж	22.12.2012	1	хлеб	23,00р.	☐
23456789	Петров	Петр	Петрович	87435454	Муж	22.12.2012	2	сок	55,00р.	☑
23456789	Петров	Петр	Петрович	87435454	Муж	22.12.2012	3	картофель	20,00р.	☐
7464798	Сидоров	Иван	Петрович	3456789	Муж	01.02.2013	1	хлеб	23,00р.	☐
7464798	Сидоров	Иван	Петрович	3456789	Муж	01.02.2013	2	сок	55,00р.	☑
7464798	Сидоров	Иван	Петрович	3456789	Муж	01.02.2013	3	картофель	20,00р.	☐
65467347	Иванова	Александра	Николаевна	75858574	Жен	11.11.2011	1	хлеб	23,00р.	☐
65467347	Иванова	Александра	Николаевна	75858574	Жен	11.11.2011	2	сок	55,00р.	☑
65467347	Иванова	Александра	Николаевна	75858574	Жен	11.11.2011	3	картофель	20,00р.	☐
11111	Иванченко	Николай	Петрович	1234565	Муж	01.01.2001	1	хлеб	23,00р.	☐
11111	Иванченко	Николай	Петрович	1234565	Муж	01.01.2001	2	сок	55,00р.	☑
11111	Иванченко	Николай	Петрович	1234565	Муж	01.01.2001	3	картофель	20,00р.	☐

Декартово произведение на языке SQL

Пример: запрос, соединяющий данные о покупателях мужского пола и товаров, на которые не действуют акции.

```
SELECT Покупатель.*,Товар.*
FROM Покупатель,Товар
WHERE Покупатель.Пол='Муж' AND Товар.Наличие_акции=FALSE;
```

Штрих-код	Название	Цена	Наличие_ак
1	хлеб	23,00р.	☐
2	сок	55,00р.	☑
3	картофель	20,00р.	☐

Номер_кар	Фамилия	Имя	Отчество	Телефон	Пол	Год_рождения
11111	Иванченко	Николай	Петрович	1234565	Муж	01.01.2001
7464798	Сидоров	Иван	Петрович	3456789	Муж	01.02.2013
23456789	Петров	Петр	Петрович	87435454	Муж	22.12.2012
65467347	Иванова	Александра	Николаевна	75858574	Жен	11.11.2011

Номер_кар	Фамилия	Имя	Отчество	Телефон	Пол	Год_рождения	Штрих-код	Название	Цена	Наличие_ак
23456789	Петров	Петр	Петрович	87435454	Муж	22.12.2012	1	хлеб	23,00р.	☐
7464798	Сидоров	Иван	Петрович	3456789	Муж	01.02.2013	1	хлеб	23,00р.	☐
11111	Иванченко	Николай	Петрович	1234565	Муж	01.01.2001	1	хлеб	23,00р.	☐
23456789	Петров	Петр	Петрович	87435454	Муж	22.12.2012	3	картофель	20,00р.	☐
7464798	Сидоров	Иван	Петрович	3456789	Муж	01.02.2013	3	картофель	20,00р.	☐
11111	Иванченко	Николай	Петрович	1234565	Муж	01.01.2001	3	картофель	20,00р.	☐

Естественное соединение

NATURAL JOIN – естественное соединение, производится по всем столбцам таблиц, имеющим одинаковые имена.

Связанными столбцами считаются те, которые в двух таблицах имеют одинаковые имена. Остальные строки из объединения исключаются.

Пример: у каких покупателей какие чеки.

```
SELECT Покупатель.*,Чек.Номер_чека
FROM Покупатель NATURAL JOIN Чек;
```

ил
и

```
SELECT Покупатель.*,Чек.Номер_чека
FROM Покупатель NATURAL JOIN Чек
WHERE Покупатель.Номер_карты=Чек.
Номер_карты;
```

Замечание: в MS ACCESS естественное соединение используется как декартово произведение при условии равенства значений в соответствующих столбцах (2 вариант)

Номер_кар	Фамилия	Имя	Отчество	Телефон	Пол	Год_рождения
11111	Иванченко	Николай	Петрович	1234565	Муж	01.01.2001
7464798	Сидоров	Иван	Петрович	3456789	Муж	01.02.2013
23456789	Петров	Петр	Петрович	87435454	Муж	22.12.2012
65467347	Иванова	Александра	Николаевна	75858574	Жен	11.11.2011

Номер_чек	Номер_карты
1	23456789
2	23456789
3	11111

Номер_кар	Фамилия	Имя	Отчество	Телефон	Пол	Год_рождения	Номер_чек
11111	Иванченко	Николай	Петрович	1234565	Муж	01.01.2001	3
23456789	Петров	Петр	Петрович	87435454	Муж	22.12.2012	1
23456789	Петров	Петр	Петрович	87435454	Муж	22.12.2012	2

Естественное соединение

Пример: вывести покупателей и товары, которые они купили.

```
SELECT Покупатель.Фамилия,Товар.Название, Товар.Наличие_акции
FROM Покупатель, Чек,  Товар_в_чеке, Товар
WHERE Покупатель.Номер_карты=Чек.Номер_карты
AND Чек.Номер_чека=Товар_в_чеке.Номер_чека
AND Товар_в_чеке.Штрих_код=Товар.Штрих_код;
```

Номер_кар	Фамилия	Имя	Отчество	Телефон	Пол	Год_рождения
11111	Иванченко	Николай	Петрович	1234565	Муж	01.01.2001
7464798	Сидоров	Иван	Петрович	3456789	Муж	01.02.2013
23456789	Петров	Петр	Петрович	87435454	Муж	22.12.2012
65467347	Иванова	Александра	Николаевна	75858574	Жен	11.11.2011

Номер_чек	Номер_карты
1	23456789
2	23456789
3	11111

Штрих_код	Номер_чек	Скидка
1	1	0
3	1	0
2	2	14
1	2	0
4	2	12
2	3	20
3	3	0

Штрих_код	Название	Цена	Наличие_ак
1	хлеб	23,00р.	☐
2	сок	55,00р.	☑
3	картофель	20,00р.	☐
4	хлеб	18,00р.	☑

Фамилия	Название	Наличие_ак
Петров	хлеб	☐
Петров	картофель	☐
Петров	сок	☑
Петров	хлеб	☐
Петров	хлеб	☑
Иванченко	сок	☑
Иванченко	картофель	☐

Внутреннее соединение

INNER JOIN – внутреннее соединение, при котором соединяются только те строки, для которых найдено совпадение значений в таблицах A и B.

Считывает результат, который включает только объединенные строки, соответствующие условиям объединения. Использует оператор сравнения (=, <>, <, <=, > или >=), чтобы сопоставить строки в двух таблицах на основании значений в общих столбцах каждой таблицы.

Замечание: в случае использования оператора = аналог естественного соединения.

Пример: у каких покупателей какие чеки.

```
SELECT Покупатель.*,Чек.Номер_чека
FROM Покупатель INNER JOIN Чек ON Покупатель.Номер_карты=Чек.Номер_карты;
```

Номер_кар	Фамилия	Имя	Отчество	Телефон	Пол	Год_рождения
11111	Иванченко	Николай	Петрович	1234565	Муж	01.01.2001
7464798	Сидоров	Иван	Петрович	3456789	Муж	01.02.2013
23456789	Петров	Петр	Петрович	87435454	Муж	22.12.2012
65467347	Иванова	Александра	Николаевна	75858574	Жен	11.11.2011

Номер_чек	Номер_карты
1	23456789
2	23456789
3	11111

Номер_кар	Фамилия	Имя	Отчество	Телефон	Пол	Год_рождения	Номер_чек
11111	Иванченко	Николай	Петрович	1234565	Муж	01.01.2001	3
23456789	Петров	Петр	Петрович	87435454	Муж	22.12.2012	1
23456789	Петров	Петр	Петрович	87435454	Муж	22.12.2012	2

Внутреннее соединение

Пример: вывести покупателей и товары, которые они купили по акции.

```
SELECT Покупатель.Фамилия,Товар.Название, Товар.Наличие_акции
FROM ((Покупатель INNER JOIN Чек ON Покупатель.Номер_карты=Чек.Номер_карты)
INNER JOIN  Товар_в_чеке ON Чек.Номер_чека=Товар_в_чеке.Номер_чека)
INNER JOIN Товар ON Товар_в_чеке.Штрих_код=Товар.Штрих_код
WHERE Товар.Наличие_акции=TRUE;
```

Номер_кар	Фамилия	Имя	Отчество	Телефон	Пол	Год_рождения
11111	Иванченко	Николай	Петрович	1234565	Муж	01.01.2001
7464798	Сидоров	Иван	Петрович	3456789	Муж	01.02.2013
23456789	Петров	Петр	Петрович	87435454	Муж	22.12.2012
65467347	Иванова	Александра	Николаевна	75858574	Жен	11.11.2011

Номер_чек	Номер_карты
1	23456789
2	23456789
3	11111

Штрих_код	Номер_чек	Скидка
1	1	0
3	1	0
2	2	14
1	2	0
4	2	12
2	3	20
3	3	0

Штрих_код	Название	Цена	Наличие_ак
1	хлеб	23,00р.	☐
2	сок	55,00р.	☑
3	картофель	20,00р.	☐
4	хлеб	18,00р.	☑

Фамилия	Название	Наличие_ак
Петров	сок	☑
Петров	хлеб	☑
Иванченко	сок	☑

Внешние соединения

Внешнее соединение считывает все строки хотя бы из одной таблицы (если они соответствуют условию поиска).

Замечание: значение имеет порядок, в котором задаются таблицы.

Левое внешнее соединение включает все строки из левой таблицы, заданные в пункте **LEFT JOIN**, а не только строки, которые совпадают для связанных столбцов. Если для строки в левой таблице нет соответствия в правой таблице, то в результате строка будет содержать NULL для всех столбцов в списке SELECT, которые были считаны из правой таблицы.

Правое внешнее соединение включает все строки из правой таблицы, заданные в пункте **RIGHT JOIN**, а не только строки, которые совпадают для связанных столбцов. Если для строки в правой таблице нет соответствия в левой таблице, то в результате строка будет содержать NULL для всех столбцов в списке SELECT, которые были считаны из левой таблицы.

Полное соединение **FULL JOIN** является комбинацией левого и

Левое соединения

Пример: вывести всех покупателей и указать, у кого какие чеки были.

SELECT Покупатель.*,Чек.Номер_чека

FROM Покупатель LEFT JOIN Чек ON Покупатель.Номер_карты=Чек.Номер_карты;

Номер_кар	Фамилия	Имя	Отчество	Телефон	Пол	Год_рождения
11111	Иванченко	Николай	Петрович	1234565	Муж	01.01.2001
7464798	Сидоров	Иван	Петрович	3456789	Муж	01.02.2013
23456789	Петров	Петр	Петрович	87435454	Муж	22.12.2012
65467347	Иванова	Александра	Николаевна	75858574	Жен	11.11.2011

Номер_чек	Номер_карты
1	23456789
2	23456789
3	11111

Номер_кар	Фамилия	Имя	Отчество	Телефон	Пол	Год_рождения	Номер_чек
11111	Иванченко	Николай	Петрович	1234565	Муж	01.01.2001	3
7464798	Сидоров	Иван	Петрович	3456789	Муж	01.02.2013	
23456789	Петров	Петр	Петрович	87435454	Муж	22.12.2012	1
23456789	Петров	Петр	Петрович	87435454	Муж	22.12.2012	2
65467347	Иванова	Александра	Николаевна	75858574	Жен	11.11.2011	

Правое соединения

Пример: вывести всех покупателей и указать, у кого какие чеки были.

```
SELECT Чек.Номер_чека, Покупатель.*
FROM Чек RIGHT JOIN Покупатель ON Покупатель.Номер_карты=Чек.Номер_карты;
```

Номер_кар	Фамилия	Имя	Отчество	Телефон	Пол	Год_рождения
11111	Иванченко	Николай	Петрович	1234565	Муж	01.01.2001
7464798	Сидоров	Иван	Петрович	3456789	Муж	01.02.2013
23456789	Петров	Петр	Петрович	87435454	Муж	22.12.2012
65467347	Иванова	Александра	Николаевна	75858574	Жен	11.11.2011

Номер_чек	Номер_карты
1	23456789
2	23456789
3	11111

Номер_чек	Номер_кар	Фамилия	Имя	Отчество	Телефон	Пол	Год_рождения
3	11111	Иванченко	Николай	Петрович	1234565	Муж	01.01.2001
	7464798	Сидоров	Иван	Петрович	3456789	Муж	01.02.2013
1	23456789	Петров	Петр	Петрович	87435454	Муж	22.12.2012
2	23456789	Петров	Петр	Петрович	87435454	Муж	22.12.2012
	65467347	Иванова	Александра	Николаевна	75858574	Жен	11.11.2011

Комбинирование строк с помощью оператора UNION

Оператор **UNION** комбинирует результаты двух запросов в один результат, который объединяет строки, считанные двумя запросами.

Выражение **UNION** удаляет из результата повторяющиеся строки. Выражение **UNION ALL** сохраняет повторы.

Ограничения:

)списки столбцов команды SELECT в двух запросах должны включать одинаковое число столбцов (названий столбцов, арифметических выражений, функций и т.д.);

)соответствующие столбцы в двух запросах должны быть заданы в одинаковом порядке;

) если имена соответствующих столбцов совпадают, то их название будет использовано в результате. Если названия соответствующих столбцов различаются, то СУБД самостоятельно определит имя столбца в результате;

)предложение ORDER BY может использоваться только в последнем запросе команды UNION;

)можно задавать предложения GROUP BY и HAVING только в отдельных запросах; их нельзя использовать для изменения конечного результата.

Комбинирование строк с помощью оператора UNION

Номер_кар	Фамилия	Имя	Отчество	Телефон	Пол	Год_рождения
11111	Иванченко	Николай	Петрович	1234565	Муж	01.01.2001
7464798	Сидоров	Иван	Петрович	3456789	Муж	01.02.2013
23456789	Петров	Петр	Петрович	87435454	Муж	22.12.2012
65467347	Иванова	Александра	Николаевна	75858574	Жен	11.11.2011

Штрих_код	Название	Цена	Наличие_ак
1	хлеб	23,00р.	☐
2	сок	55,00р.	☑
3	картофель	20,00р.	☐
4	хлеб	18,00р.	☑

```
SELECT Покупатель.Фамилия
FROM Покупатель
UNION
SELECT Товар.Название
FROM Товар;
```

Фамилия
Иванова
Иванченко
картофель
Петров
Сидоров
сок
хлеб

```
SELECT Покупатель.Фамилия
FROM Покупатель
UNION ALL
SELECT Товар.Название
FROM Товар;
```

Фамилия
Петров
Сидоров
Иванова
Иванченко
хлеб
сок
картофель
хлеб

Подзапросы

Подзапрос – это команда SELECT, встроенная в другую команду SQL.

Отличия структуры подзапросов:

1) подзапрос можно поместить в предложение SELECT, FROM, WHERE, HAVING или в другой запрос;

2) подзапрос всегда должен заключаться в круглые скобки;

3) подзапрос нельзя заканчивать точкой с запятой;

4) не помещайте в подзапрос предложение ORDER BY;

5) подзапрос включает одну команду SELECT;

6) подзапрос может использовать столбцы в таблицах, которые приводятся в предложении FROM самого подзапроса или другого подзапроса;

7) если таблица появляется во внутреннем, а не во внешнем запросе, вы не сможете включить столбцы этой таблицы в конечный результат (то есть в предложение SELECT внешнего запроса);

7) в зависимости от ситуации подзапрос может использоваться для считывания ограниченного количества строк или столбцов. Подзапрос может считывать пустую таблицу

Подзапросы

Чаще всего подзапросы используются в предложении WHERE в одной из форм:

1) WHERE условие_поиска = (подзапрос);
2) WHERE условие_поиска [NOT] IN (подзапрос);
3) WHERE условие_поиска = ALL (подзапрос);
4) WHERE условие_поиска = ANY (подзапрос);
5) WHERE [NOT] EXISTS (подзапрос).

Замечание: большую часть подзапросов можно записать в виде объединений.

Подзапросы

Пример: вывести всех покупателей, у которых имеются чеки.

```
SELECT Покупатель.*
FROM Покупатель
WHERE Покупатель.Номер_карты IN (SELECT DISTINCT Чек.Номер_карты
                                 FROM Чек);
```

Номер_кар	Фамилия	Имя	Отчество	Телефон	Пол	Год_рождения
11111	Иванченко	Николай	Петрович	1234565	Муж	01.01.2001
7464798	Сидоров	Иван	Петрович	3456789	Муж	01.02.2013
23456789	Петров	Петр	Петрович	87435454	Муж	22.12.2012
65467347	Иванова	Александра	Николаевна	75858574	Жен	11.11.2011

Номер_чек	Номер_карты
1	23456789
2	23456789
3	11111

Номер_кар	Фамилия	Имя	Отчество	Телефон	Пол	Год_рождения
23456789	Петров	Петр	Петрович	87435454	Муж	22.12.2012
11111	Иванченко	Николай	Петрович	1234565	Муж	01.01.2001

Подзапросы

Пример: вывести товары, цена которых больше средней величины цен по всем товарам по акции.

```
SELECT Товар.Название, Товар.Цена,Товар.Наличие_акции
FROM Товар
WHERE Товар.Цена>(SELECT AVG(Товар.Цена)
FROM Товар
WHERE Товар.Наличие_акции=TRUE);
```

Штрих-код	Название	Цена	Наличие_ак
1	хлеб	23,00р.	☐
2	сок	55,00р.	☑
3	картофель	20,00р.	☐
4	хлеб	30,00р.	☐
5	сок	70,00р.	☐
6	хлеб	22,00р.	☑
7	сок	58,00р.	☑

Название	Цена	Наличие_ак
сок	55,00р.	☑
сок	70,00р.	☐
сок	58,00р.	☑

Подзапросы

Пример: вывести товары, цена которых больше минимальной цены по всем товарам без акции, но меньше максимальной цены по всем товарам по акции.

```
SELECT Товар.Название, Товар.Цена,Товар.Наличие_акции
FROM Товар
WHERE Товар.Цена BETWEEN (SELECT MIN(Товар.Цена)
FROM Товар
WHERE Товар.Наличие_акции=FALSE) AND (SELECT MAX(Товар.Цена)
FROM Товар
WHERE Товар.Наличие_акции=TRUE);
```

Штрих-код	Название	Цена	Наличие_ак
1	хлеб	23,00р.	☐
2	сок	55,00р.	☑
3	картофель	20,00р.	☐
4	хлеб	30,00р.	☐
5	сок	70,00р.	☐
6	хлеб	22,00р.	☑
7	сок	58,00р.	☑

Название	Цена	Наличие_ак
хлеб	23,00р.	☐
сок	55,00р.	☑
картофель	20,00р.	☐
хлеб	30,00р.	☐
хлеб	22,00р.	☑
сок	58,00р.	☑