# Лекция 3

## Основы языка SQL
## (Structured Query Language)

# Язык запросов SQL

Для получения сведений из БД пользователь дает СУБД ***запрос***. СУБД обрабатывает его и отправляет пользователю результат.

Запросы к СУБД формулируются на специальных ***языках запросов***. С их помощью указывается какие данные надо получить, но не задается способ выполнения в отличие от языков программирования.

Фактическим стандартом языка запросов для современных реляционных СУБД является ***SQL*** (***Structured Query Language***) версий 2 и 3 - ***SQL2, SQL3***.

Запросы на языке SQL строятся с использованием операторов

| | |
|---|---|
| **SELECT** | Выбрать строки |
| **INSERT** | Вставить строку |
| **UPDATE** | Обновить строку |
| **DELETE** | Удалить строки |
| **COMMIT** | Завершить транзакцию |
| **ROLLBACK** | Откатить транзакцию |
| **CREATE TABLE** | Создать таблицу |
| **CREATE INDEX** | Создать индекс |

Результат выполнения запроса формируется в виде таблицы.

# Формы языка SQL

- **Интерактивный** SQL. позволяет конечному пользователю в интерактивном режиме выполнять SQL-операторы. Все СУБД предоставляют инструментальные средства для работы с базой данных в интерактивном режиме. Например, СУБД Oracle включает утилиту SQL*Plus, позволяющую в строчном режиме выполнять большинство SQL-операторов
- **Статический** SQL. может реализовываться как встроенный SQL или модульный SQL. Операторы статического SQL определены уже в момент компиляции программы
- **Динамический** SQL. позволяет формировать операторы SQL во время выполнения программы
- **Встроенный** SQL. позволяет включать операторы SQL в код программы на другом языке программирования (например, C++).

# Группы операторов SQL

Язык SQL определяет:

- операторы языка, называемые иногда командами языка SQL;
- типы данных;
- набор встроенных функций.

По своему логическому назначению операторы языка SQL часто разбиваются на следующие группы:

- язык определения данных DDL (Data Definition Language):
  - CREATE TABLE, ALTER TABLE, DROP TABLE, CREATE INDEX, ALTER INDEX, DROP INDEX
- язык манипулирования данными DML (Data Manipulation Language):
  - SELECT - извлечение данных из одной или нескольких таблиц;
  - INSERT - добавление строк в таблицу;
  - DELETE - удаление строк из таблицы;
  - UPDATE - изменение значений полей в таблице.

# Фазы выполнения SQL-оператора

**SELECT A,B,C, FROM X,Y WHERE A<500 AND C='ASF'**

**parse** - Синтаксический разбор оператора

**validate** - Проверка привилегий пользователя,проверка действительности имен системных каталогов, таблиц и названий полей

**access plan** - Генерация плана доступа к ресурсам. План доступа - это двоичное представление выполнимого кода по отношению к данным, сохраняемым в БД

**optimize** - Оптимизация плана доступа. Для увеличения скорости поиска данных могут применяться индексы. Оптимизация использования взаимосвязанных таблиц

**execute** - Выполнение оператора

# Примеры языка манипулирования данными

**CREATE**

```
CREATE TABLE student
(St_id int NOT NULL PRIMARY KEY,
St_name varchar(45) NOT NULL,
Speciality int(10) not null);
```

**INSERT**

```
insert into student (St_id, St_name, Speciality) values
        (1, "Иванов", 2), (2, "Петров", 1);
```

**UPDATE**

```
UPDATE имя_таблицы SET
имя_столбца_1=значение, ..., имя_столбца_n=значение
[WHERE УСЛОВИЕ];
```

**DELETE**

```
DELETE FROM имя_таблицы
        [WHERE условие_удаления]
```

# Создание таблицы (Пример)

```
CREATE TABLE attainment(
    At_id    int(3) NOT NULL PRIMARY KEY,
    St_id    int(3) NOT NULL,
    Sub_id   int(3) NOT NULL,
    T_id     int(3) NOT NULL,
    Date     date,
    Rating   int(4) NOT NULL,
    FOREIGN KEY(St_id) REFERENCES student(St_id),
    FOREIGN KEY(Sub_id) REFERENCES subject(Sub_id),
    FOREIGN KEY(T_id) REFERENCES teacher(T_id));
```

# Оператор SELECT



Оператор выбора строк **SELECT** реализует ***все операции*** реляционной алгебры и имеет достаточно сложный синтаксис.

Самый простой запрос содержит только два обязательных ***ключевых слова***, которые включает в себя запрос любой сложности: **SELECT** и **FROM:**

```
SELECT *
FROM Улицы, ТипыБолезней
```

Общий синтаксис запроса:

```
SELECT <имена полей (что)>
FROM <имена таблиц (откуда)>
```

Символ * (звезда) используется для выбора всех полей исходных таблиц.

Таким способом выполняется реляционная операция «Декартово произведение отношений».


| Улицы | |
|---|---|
| **КодУли** | **НаимУл** |
| 1 | Вешняк |
| 2 | Юности |

| ТипыБолезней | |
|---|---|
| **КодТипаБ** | **ТипБоле** |
| 1 | Сердечно |
| 2 | Опорнодв |

| Результат | | | |
|---|---|---|---|
| **КодУли** | **НаимУл** | **КодТипаБ** | **ТипБоле** |
| 1 | Вешняк | 1 | Сердечно |
| 1 | Вешняк | 2 | Опорнодв |
| 2 | Юности | 1 | Сердечно |
| 2 | Юности | 2 | Опорнодв |

# Обобщенный формат оператора ALTER TABLE

**ALTER TABLE имя_таблицы**

**[ADD [COLUMN] имя_столбца тип_данных** [ NOT NULL ][UNIQUE]

[DEFAULT <значение>][ CHECK (<условие_выбора>)]]

**[DROP [COLUMN] имя_столбца** [RESTRICT | CASCADE ]]

**[ADD [CONSTRAINT [имя_ограничения]]**

**[{PRIMARY KEY (имя_столбца [,...n]) |[UNIQUE (имя_столбца [,...n])}**

**|[FOREIGN KEY (имя_столбца_внешнего_ключа [,...n]) REFERENCES имя_род_таблицы [(имя_столбца_род_таблицы [,...n])],**

[ MATCH {PARTIAL | FULL} [ON UPDATE {CASCADE| SET NULL | SET DEFAULT | NO ACTION}] [ON DELETE {CASCADE| SET NULL | SET DEFAULT | NO ACTION}] |[CHECK(<условие_выбора>)][,...n]}]

**[DROP CONSTRAINT имя_ограничения** [RESTRICT | CASCADE]]

[ALTER [COLUMN] SET DEFAULT <значение>]

[ALTER [COLUMN] DROP DEFAULT]

# Обобщенный оператор SELECT

**(сокращенный вариант)**

```
SELECT столбцы
FROM таблица
 [WHERE условие_фильтрации_строк]
 [GROUP BY столбцы_для_группировки]
 [HAVING условие_фильтрации_групп]
 [ORDER BY столбцы_для_сортировки]
```

# Получение проекций

Для получения проекций результата выполнения реляционных операций достаточно в операторе выбора строк **SELECT** указать имена полей, включаемых в таблицу-результат.

```
SELECT НаимДолжн, Оклад
FROM Должности
```

| Должности |
|---|
| **КодДолжн** |
| НаимДолжн |
| Оклад |

| Результат |
|---|
| НаимДолжн |
| Оклад |

```
SELECT Улицы.НаимУлицы,
    ТипыБолезней.ТипБолезни
FROM Улицы, ТипыБолезней
```

```
SELECT НаимУлицы,
    ТипБолезни
FROM Улицы, ТипыБолезней
```

| Улицы |
|---|
| **КодУлицы** |
| НаимУлиц |

| ТипыБолезней |
|---|
| **КодТипаБол** |
| ТипБолезни |

| Результат |
|---|
| НаимУлицы |
| ТипБолезни |

| Улицы | |
|---|---|
| **КодУли** | **НаимУл** |
| 1 | Вешняк |
| 2 | Юности |

| ТипыБолезней | |
|---|---|
| **КодТипаБ** | **ТипБоле** |
| 1 | Сердечно |
| 2 | Опорнодв |

| Результат | |
|---|---|
| **НаимУл** | **ТипБоле** |
| Вешняк | Сердечно |
| Вешняк | Опорнодв |
| Юности | Сердечно |
| Юности | Опорнодв |

# Исключение повторений

Для исключения строк-дубликатов из таблицы-результата надо после ключевого слова **SELECT** указать ключевое слово **DISTINCT**.

| Участки |
|---|
| **КодАдреса** |
| КодУлицы |
| НомДома |
| НомКорп |
| НомУчастка |

Участки

| КодАдреса | КодУлицы | НомДома | НомКорп | НомУчастка |
|---|---|---|---|---|
| 1 | 1 | 4 | 1 | 2 |
| 2 | 2 | 2 | 2 | 2 |
| 3 | 6 | 9 | 2 | 1 |
| 4 | 2 | 2 | 1 | 2 |
| 5 | 3 | 7 | | 3 |
| 6 | 6 | 9А | | 1 |
| 7 | 4 | 3 | 1 | 3 |

***Запрос.*** Выдать перечень обслуживаемых поликлиникой участков.

```
SELECT НомУчастка
FROM Участки
```

Результат

| НомУчастка |
|---|
| 2 |
| 2 |
| 1 |
| 2 |
| 3 |
| 1 |
| 3 |

```
SELECT DISTINCT НомУчастка
FROM Участки
```

Результат

| НомУчастка |
|---|
| 2 |
| 1 |
| 3 |

# Отбор записей из таблиц

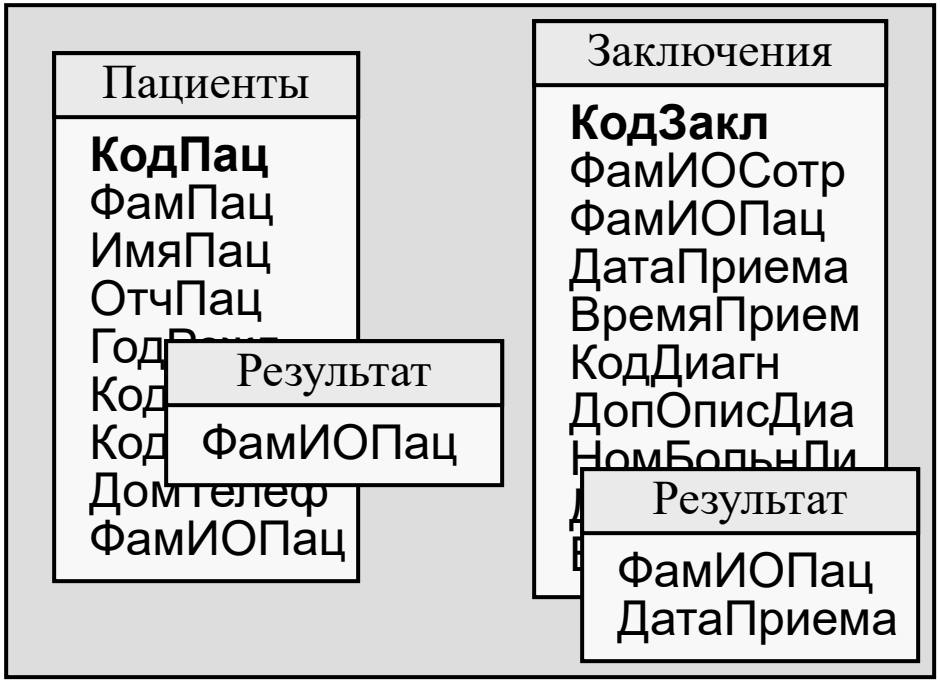
Условия отбора записей из таблиц задаются в операторе **SELECT** с помощью ***предикатов*** после ключевого слова **WHERE**

Общий синтаксис запроса:

**SELECT <имена полей>**
**FROM <имена таблиц>**
**WHERE <предикат (условие)>**

***Запрос.*** Выдать список пациентов с годом рождения до 1942.

```
SELECT DISTINCT ФамИОПац
FROM Пациенты
WHERE ГодРожд<=1941
```




***Запрос.*** Выдать список пациентов с датой приема, посетивших поликлинику в августе 2001 года в первой половине рабочего дня.

```
SELECT DISTINCT ФамИОПац, ДатаПриема
FROM Заключения
WHERE ДатаПриема BETWEEN 01.08.01 AND 31.08.01
   AND ВремяПриема < 13:30
```

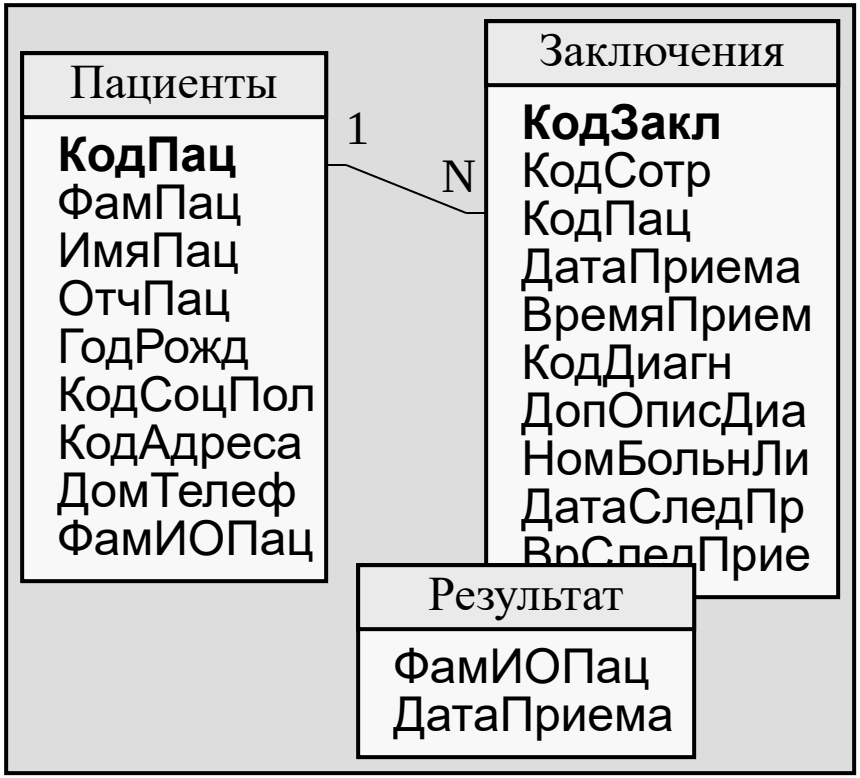
# Отбор записей из нескольких таблиц

Отбор записей из нескольких таблиц реализует реляционную операцию «Соединение».

После ключевого слова **WHERE** в качестве условия соединения таблиц размещают предикаты сравнения значений их ключевых полей.

***Запрос.*** Выдать список пациентов с датами посещения поликлиники, назначенных на повторный прием.

```
SELECT DISTINCT ФамИОПац,
    ДатаПриема
FROM Пациенты, Заключения
WHERE Пациенты.КодПац=
    Заключения.КодПац
    AND ДатаСледПосещ IS NOT NULL
```




Если значение поля в некоторой записи не задано, оно считается ***неопределенным*** и обозначается **NULL**.

Предикаты для проверки на неопределенность: **IS NULL** и **IS NOT NULL**.

# Операции с неопределенным значением

***Неопределенное значение*** в теории БД интерпретируется как значение, неизвестное на данный момент времени.

Для неопределенных значений, обозначаемых **NULL**, не действуют стандартные правила сравнения: два неопределенных значения всегда не равны друг другу.

Введение неопределенного значения потребовало расширить в рамках теории БД классическую двузначную логику до трехзначной.

**Таблица истинности логических операций**

| A | B | NOT A | A AND B | A OR B |
|---|---|---|---|---|
| TRUE | TRUE | FALSE | TRUE | TRUE |
| TRUE | FALSE | FALSE | FALSE | TRUE |
| TRUE | NULL | FALSE | NULL | TRUE |
| FALSE | TRUE | TRUE | FALSE | TRUE |
| FALSE | FALSE | TRUE | FALSE | FALSE |
| FALSE | NULL | TRUE | FALSE | NULL |
| NULL | TRUE | NULL | NULL | TRUE |
| NULL | FALSE | NULL | FALSE | NULL |
| NULL | NULL | NULL | NULL | NULL |

# Группировка данных в таблицах

Для анализа данных с одинаковыми значениями атрибутов записи таблицы разбиваются на группы. Интересующие значения указываются после ключевого слова **GROUP BY**.

В результате для полученных групп записей можно определить некоторые характерные значения с помощью ***агрегатных функций***, вычисляющих одно значение для каждой группы:

| | |
|---|---|
| **COUNT** | Количество записей |
| **SUM** | Сумма значений |
| **AVG** | Среднее значение |
| **MIN** | Минимальное значение |
| **MAX** | Максимальное значение |

***Запрос.*** Сколько участков обслуживает поликлиника (ск.разных номеров участков).

```
SELECT COUNT (DISTINCT НомУчастка)
FROM Участки
GROUP BY НомерУчастка
```

Участки

| КодАдреса |
|---|
| КодУлицы |
| НомДома |
| НомКорп |
| НомУчастка |

Результат

| НомУчастка | COUNT (*) |
|---|---|
| 2 | 3 |
| 1 | 2 |
| 3 | |

Результат

| COUNT (DI |
|---|
| 3 |

# Пример сложного запроса

***Запрос.*** Какие отделения поликлиники и какое количество пенсионеров 2-го участка с диагнозом «Радикулит» посетило за осенне-зимний период 2000-01гг?


```
SELECT DISTINCT Подразделения.НаимПодр,
    COUNT(DISTINCT Участки. НомУчастка)
FROM Подразделения
W
    WHERE Подразделения.КодПодр = Сотрудники.КодПодр
        AND Сотрудники.КодСотр = Заключения.КодСотр
        AND Заключения. КодПац = Пациенты.КодПац
        AND Пациенты.КодСоцПол = СоцПол.КодСоцПол
        AND Пациенты.КодАдреса = Участки.КодАдреса
GROUP BY Подразделения. НаимПодр
```