# Язык SQL

- *исторические аспекты развития SQL;*
- *структура и типы данных языка SQL;*
- *операторы языка SQL:*
  - *операторы манипулирования данными,*
  - *операторы определения данных.*

***SQL*** *– это язык программирования, применяемый для создания, модификации, поиска и извлечения информации, хранящейся в произвольной реляционной базе данных, управляемой соответствующей системой управления базами данных (СУБД).*


- ***непроцедурный;***
- ***встроенный или интерактивный;***
- ***стандартизованный;***
- ***используемый для управления данными и объектами баз данных.***

*В **SQL** определены два подмножества языка:*

- ***SQL-DDL** (Data Definition Language) — язык определения данных. Сюда относятся команды создания и удаления баз данных; создания, изменения и удаления таблиц; управления пользователями и т.д.*
- ***SQL-DML** (Data Manipulation Language) — язык манипулирования (управления) данными: добавление, изменение, удаление и извлечение данных, управления транзакциями.*

# Операторы языка SQL

| Вид | Название | Назначение |
|---|---|---|
| DDL | CREATE TABLE | создание таблицы |
| | DROP TABLE | удаление таблицы |
| | ALTER TABLE | изменение структуры таблицы |
| | CREATE INDEX | создание индекса |
| | DROP INDEX | удаление индекса |
| | CREATE VIEW | создание представления |
| | DROP VIEW | удаление представления |
| | GRAND | назначение привилегий |
| | REVOKE | удаление привилегий |
| DML | SELECT | выборка записей |
| | UPDATE | изменение записей |
| | INSERT | вставка новых записей |
| | DELETE | удаление записей |

## ❑ *Оператор создания таблицы* ***CREATE TABLE***

CREATE TABLE <имя таблицы>
(<имя столбца> <тип данных> [NOT NULL]
[,<имя столбца> <тип данных> [NOT NULL]]... )

**Пример** создания таблицы.

Пусть требуется создать таблицу goods описания товаров, имеющую поля: type — вид товара, comp_id — идентификатор компании-производителя, name — название товара и price — цена товара. Оператор определения таблицы может иметь следующий вид:

**CREATE TABLE** goods (type SQL_CHAR(8) NOT NULL, comp_id SQL_CHAR(10) NOT NULL, name SQL_VARCHAR(20), price SQL_DECIMAL(8,2)).

- *Оператор изменения структуры таблицы* ***ALTER TABLE***

ALTER TABLE <имя таблицы>
( {ADD, MODIFY, DROP} <имя столбца> [<тип данных>]
[NOT NULL]
[,{ADD, MODIFY, DROP} <имя столбца> [<тип данных>]
[NOT NULL]]...)

***ADD*** *добавление;*
***MODIFY*** *изменение;*
***DROP*** *удаление.*

**Пример** добавление поля таблицы.

Пусть в созданной ранее таблице goods необходимо добавить поле number, отводимое для хранения величины запаса товара. Для этого следует записать оператор вида:

**ALTER TABLE** goods (ADD number SQL_INTEGER).

## ❑ *Оператор создания индекса* ***CREATE INDEX***

CREATE [UNIQUE] INDEX <имя индекса>
ON <имя таблицы>
(<имя столбца> [ ASC | DESC ]
[,<имя столбца> [ ASC | DESC ]... )

***ASC*** *сортировка по возрастанию*
***DESC*** *сортировка по убыванию*

**Пример** создания индекса.
Пусть для таблицы EMP, имеющей поля: NAME (имя), SAL (зарплата), MGR (руководитель) и DEPT (отдел), нужно создать индекс main_indx для сортировки имен в алфавитном порядке и убыванию размеров зарплаты.
Оператор создания индекса может иметь вид:
**CREATE INDEX** main_indx
ON emp (name, sal DESC).

## ❑ *Оператор удаления таблицы* ***DROP TABLE***

DROP TABLE <имя таблицы>

**Пример** удаления таблицы.
Для удаления таблицы с именем items достаточно записать оператор вида:
**DROP TABLE** items.

## ❑ *Оператор удаления индекса* ***DROP INDEX***

DROP INDEX <имя индекса>

**Пример** удаления индекса.
Для уничтожения индекса main_indx к таблице emp достаточно записать оператор:
**DROP INDEX** main_indx.

## ❑ *Оператор формирования запросов* ***SELECT***

SELECT [ALL | DISTINCT]
<список данных>
FROM <список таблиц>
[WHERE <условие выборки>]
[GROUP BY <имя столбца> [,<имя столбца>] ... ]
[HAVING <условие поиска>]
[ORDER BY <спецификация> [,<спецификация>] ...]

***DISTINCT*** *исключение повторяющихся строк*

***GROUP BY*** *группировка по списку*

***ORDER BY*** *задает порядок сортировки*

**Пример** *выбора записей.*

Для таблицы EMP, имеющей поля: NAME (имя), SAL (зарплата), MGR (руководитель) и DEPT (отдел), требуется вывести имена сотрудников и размер их зарплаты, увеличенный на 100 единиц. Оператор выбора можно записать следующим образом:

```
SELECT name, sal+100
FROM emp.
```

**Пример** *выбора с условием.*

Вывести названия таких отделов таблицы EMP, в которых в данный момент отсутствуют руководители. Оператор SELECT имеет вид:

```
SELECT dept
FROM emp
WHERE mgr is NULL.
```

**Пример** *выбора с группированием.*

Пусть требуется найти минимальную и максимальную зарплаты для каждого из отделов (по таблице EMP). Оператор SELECT для этого запроса имеет вид:

```
SELECT dept, MIN(sal), MAX(sal)
FROM emp
GROUP BY dept.
```

## ❑ *Предложение **ORDER BY***

**В общем случае строки в результирующей таблице SQL- запроса никак не упорядочены. Однако их можно требуемым образом отсортировать, для чего в оператор SELECT помещается фраза ORDER BY, которая сортирует данные выходного набора в заданной последовательности.**

**ASC – сортировка по возрастанию (реализуется по умолчанию);**
**DESC – сортировка по убыванию.**

**! Фраза ORDER BY всегда должна быть последним элементом в операторе SELECT**

## ❑ *Оператор изменения записей* ***UPDATE***

UPDATE <имя таблицы>
SET <имя столбца> = {<выражение> , NULL }
[, SET <имя столбца> = {<выражение> , NULL }... ]
[WHERE <условие>]

**Пример** *изменения записей.*

Пусть необходимо увеличить на 500 единиц зарплату тем служащим, которые получают не более 6000 (по таблице EMP). Запрос, сформулированный с помощью оператора SELECT, может выглядеть так:

**UPDATE** emp
SET sal = 6500
WHERE sal <= 6000.

- *Оператор вставки новых записей* ***INSERT INTO***

| | |
|---|---|
| INSERT INTO <имя таблицы> [(<список столбцов>)] VALUES (<список значений>) | INSERT INTO <имя таблицы> [(<список столбцов>)] <предложение SELECT> |

**Пример** *ввода записей.*

Ввести в таблицу EMP запись о новом сотруднике. Для этого можно записать такой оператор вида:

**INSERT INTO** emp
VALUES («Ivanov», 7500, «Lee», «cosmetics»).

## ❑ *Оператор удаления записей* ***DELETE FROM***

DELETE FROM <имя таблицы>
[WHERE <условие>]

**Пример** *удаления записей.*

В связи с ликвидацией отдела игрушек (toy), требуется удалить из таблицы EMP всех сотрудников этого отдела. Оператор DELETE для этой задачи будет выглядеть так:

**DELETE FROM** emp
WHERE dept = «toy».

- *Оператор создания представления* ***CREATE VIEW***

CREATE VIEW <имя представления>
[(<имя столбца> [,<имя столбца> ]...)]
AS <оператор SELECT>

**Пример** *создание представления.*

Пусть имеется таблица companies описания производителей товаров с полями: comp_id (идентификатор компании), comp_name (название организации), comp_address (адрес) и phone (телефон), а также таблица goods производимых товаров с полями: type (вид товара), comp_id (идентификатор компании), name (название товара) и price (цена товара). Таблицы связаны между собой по полю comp_id. Требуется создать представление repr с краткой информацией о товарах и их производителях: вид товара, название производителя и цена товара. Оператор определения представления может иметь следующий вид:

**CREATE VIEW**

```
герг
AS
SELECT
goods.type, companies.comp_name, goods.price
FROM
goods, companies
WHERE
goods.comp_id = companies.comp_id
```

- *Оператор удаления представления* ***DPOP VIEW***

DROP VIEW <имя представления>

**Пример** *удаления представления.*

Удаление представления герг производится оператором вида:

DROP VIEW герг.

## ❑ *Понятие подзапроса*

**! Подзапрос – это инструмент создания временной таблицы, содержимое которой извлекается и обрабатывается внешним оператором. Текст подзапроса должен быть заключен в скобки.**

**К подзапросам применяются следующие правила и ограничения:**

- **Фраза ORDER BY не используется, хотя и может присутствовать во внешнем подзапросе;**
- **Список в предложении SELECT состоит из имен отдельных столбцов или составленных из них предложений – за исключением случая, когда в запросе присутствует ключевое слово EXISTS;**
- **По умолчанию имена столбцов в подзапросе относятся к таблице, имя которой указано в предложении FROM**
- **Если подзапрос является одним из двух операндов, участвующих в операции сравнения, то запрос должен указываться в правой части этой операции.**

# Существует два типа подзапросов:

- **Скалярный подзапрос** возвращает единственное значение;
- **Табличный подзапрос** возвращает множество значений, т.е. значения одного или нескольких столбцов таблицы, размещенные в более чем одной строке.

**Пример** *использования подзапросов.*

Определить по какому предмету сдается последний экзамен. Оператор SELECT для этой задачи будет выглядеть так:

```
SELECT subj_name, date FROM exams, subject
WHERE date= (SELECT max (dat) FROM exams)
```

## ❑ *Использование ключевых слов ANY и ALL*

**! ALL – условие считается выполненным, только когда оно выполняется для всех значений в результирующем столбце подзапроса.**

**! ANY – условие считается выполненным, когда оно выполняется хотя бы для одного из значений в результирующем столбце подзапроса.**

**Пример** *использования ключевого слова ALL.*

Выбрать студентов-отличников (при условии, что у каждого студента есть хотя бы одна оценка)

```
SELECT s.surname, s.name, s.patron FROM student s
WHERE 5= ALL (SELECT mark FROM rating r WHERE s.stud_id=r.stud.id)
```