Базы данных.
Язык SQL

Зачем нужен язык запросов?

- Операции:
  - над множествами (объединение, пересечение, разность, деление, декартово произведение) над отношениями

Основы SQL: SQL?

- Structured Query Language — язык структурированных запросов
- Информационно-логический язык, а не язык программирования!
- Для создания, модификации и управления данными в РБД.

Основы SQL: синтаксис

Операторы состоят из:

- имен операций и функций;
- имен таблиц и их столбцов;
- зарезервированных ключевых слов и специальных символов;
- логических, арифметических и строковых выражений.

Основы SQL: синтаксис

Общий вид простого оператора:

```
ГЛАГОЛ параметры;
```

Пример:

```
SELECT `id` FROM `mytable`;
```

Основы SQL: синтаксис

- Выражения не зависят от регистра, не требуют наличия кавычек...

- «Правильный» стиль:

```
INSERT INTO `news` (`id`,`post_date`)
  VALUES (42, '2008-06-01 04:13:15');
```

Основы SQL: синтаксис

Комментарии:

```
-- Структура таблицы

/*
Версия сервера: 5.0.54
Дамп базы данных от: 01/01/12
*/
```

Основы SQL: типы данных

Целые числа:

- **TINYINT** — 1 байт, -128..127 (0..255);
- **SMALLINT** — 2 байта, -32768..32767;
- **MEDIUMINT** — 3 байта;
- **INT** — 4 байта;
- **BIGINT** — 8 байт.

Основы SQL: типы данных

Дробные числа:

- **FLOAT** (4 байта);
- **DOUBLE** (8 байт).

Основы SQL: типы данных

Строки:

- **CHAR** — дополняет до «ширины»;
- **VARCHAR** — использует минимум;
- **(TINY|MEDIUM|LONG)BLOB** — бинарные данные;
- **TEXT** — текстовые данные;
- **ENUM** — одно из значений;
- **SET** — ноль или более значений.

Основы SQL: типы данных

Другие:

- **BOOL**, **BOOLEAN**;
- **SERIAL** — BIGINT UNSIGNED NOT NULL AUTO_INCREMENT UNIQUE;
- **DATETIME**, **DATE**, **TIMESTAMP**, **TIME**, **YEAR**.

Основы SQL: команды

Базы данных: создание

```
CREATE DATABASE `db_name`;
```

(!) *Большинство SQL СУБД позволяют создавать БД с кириллическими и специальными символами в названии, но рекомендуется использовать только символы латинского алфавита, цифры и знак «_».*

Основы SQL: команды

Базы данных: удаление

```
DROP DATABASE `db_name`;
```

Базы данных: смена текущей

```
USE `db_name`;
```

Основы SQL: команды

Таблицы: создание

```
CREATE TABLE `table_name` (
   /*описание полей таблицы*/
`название_поля` тип параметры,
`название_поля` тип,
`название_поля` тип параметры
);
```

Основы SQL: команды

Таблицы: создание

Пример:

```
CREATE TABLE `news` (
    `id` mediumint(8) UNSIGNED
  PRIMARY KEY NOT NULL
AUTO_INCREMENT,
    `posted` TIMESTAMP NOT NULL,
    `content` TEXT,
);
```

Создание таблицы из набора данных

```
Select <список столбцов |выражений>
  Into <имя новой таблицы>
From <таблица | запрос>
[Where <логическое выражение>]
[Order By <столбец | выражение>];

Select Kdf, Nazf, Izd, Kdizd, God,
  Round((z1+z2+z3+z4)/4,2) As Sred
Into Nfirm
From Firm
Where Izd In("телевизор","телефон");
```

Основы SQL: команды

Таблицы: модификации

Переименование

```
ALTER TABLE `table_name` RENAME
 TO `table_name2`;
```

Основы SQL: команды

Таблицы: модификации

Добавление столбца

```
ALTER TABLE `table_name`
   ADD COLUMN `new_column`
   описание_столбца
    [FIRST | AFTER`after_column`];
```

Основы SQL: команды

Таблицы: модификации

Модификация столбца

```
ALTER TABLE `table_name`
  MODIFY COLUMN `bad_column`
  описание_столбца;
```

Основы SQL: команды

Таблицы: модификации

Удаление столбца

```
ALTER TABLE `table_name`
   DROP COLUMN
   `very_bad_column`;
```

Основы SQL: команды

Таблицы: удаление

```
DROP TABLE `table_name`;
```

Основы SQL: команды

Строки: добавление (вставка)

```
INSERT INTO `table_name`
  [ (`field1`,`field2`, ...,
 `fieldN`) ]
  VALUES ('value1',
  'value2', ..., `valueN`);
```

Основы SQL: команды

Строки: модификация (обновление)

```
UPDATE table_name
   SET field1 = value1,
     field2 = value2, ...,
     fieldN = valueN
   [ WHERE условие ];
```

Основы SQL: команды

Строки: удаление

```
DELETE FROM `table_name`
[ WHERE условие ];
```

Основы SQL: выборка

```
SELECT field1,..., fieldN
FROM table1,..., tableN
[ WHERE условие ]
[ GROUP BY field1,..., fieldN ]
[ ORDER BY field1,..., fieldN
  [ ASC | DESC ] ]
```

Основы SQL: выборка

```
SELECT * FROM table_name;
+------+------+
| num  | id   |
+------+------+
|    1 |    1 |
|    1 |    2 |
|    3 |    2 |
|    1 |    7 |
|    2 |    1 |
|    1 |   42 |
+------+------+
```

Основы SQL: выборка

WHERE

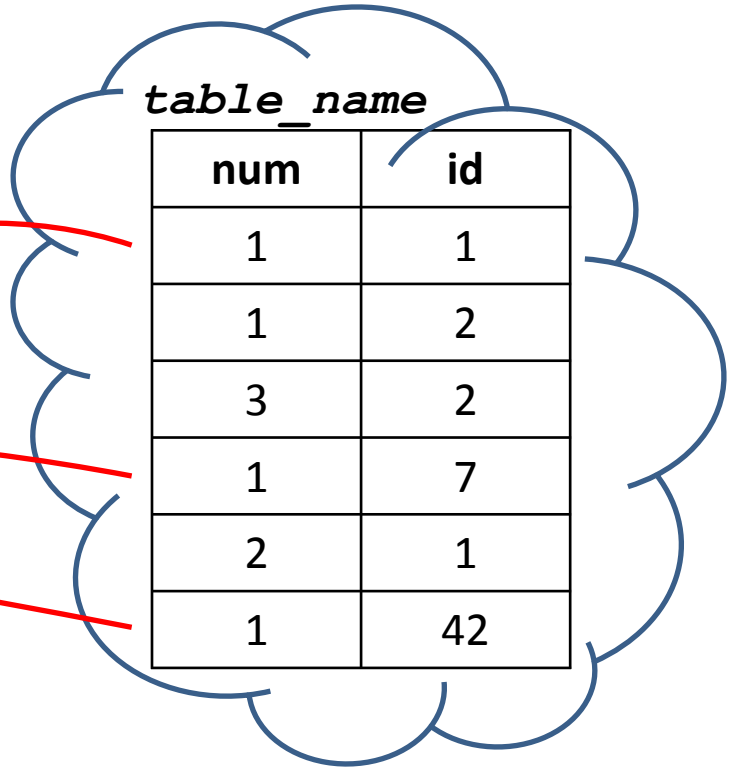
- Условный оператор
- Используется для отбора записей
- Служит параметром в выражениях с SELECT, DELETE, UPDATE
- Действует на исходный набор записей (до группировки)*

*Для отбора записей после группировки используется **HAVING**

Основы SQL: выборка

```
SELECT * FROM table_name
  WHERE num = 1 AND (id > 5 OR id
 < 2 );
```

```
+------+------+
| num  | id   |
+------+------+
|    1 |    1 |
|    1 |    7 |
|    1 |   42 |
+------+------+
```



Основы SQL: выборка

ORDER BY

- Определяет порядок сортировки
- Добавляется после **WHERE**
- Для каждого параметра может быть указано направление сортировки (**ASC, DESC**)
- Каждый следующий параметр определяет сортировку записей внутри группы с одинаковым значением предыдущего

Основы SQL: выборка

```
SELECT * FROM `table_name`
  ORDER BY `num` ASC, `id` DESC;
+------+------+
| num  | id   |
+------+------+
|    1 |   42 |
|    1 |    7 |
|    1 |    2 |
|    1 |    1 |
|    2 |    1 |
|    3 |    2 |
+------+------+
```

Основы SQL: выборка

LIMIT

- Ограничивает число строк в выдаче (результате выборки)
- Размещается в самом конце запроса
- Разрешает начинать выдачу с нужной строки результирующего набора записей
  (**LIMIT [offset], rows**)

Основы SQL: выборка

```
SELECT * FROM table_name
   LIMIT 2;
+------+------+
| num  | id   |
+------+------+
|    1 |    1 |
|    1 |    2 |
+------+------+
```

Основы SQL: выборка

DISTINCT

- Ограничивает набор обрабатываемых строк, разрешая только строки с неповторяющимися значениями заданных параметров
- Размещается в блоке **SELECT** перед выбранными параметрами

Основы SQL: выборка

```
SELECT DISINCT num
FROM table_name;
+------+
| num  |
+------+
|    1 |
|    2 |
|    3 |
+------+
```

Основы SQL: выборка

GROUP BY

- Группирует строки для выполнения групповых операций
- Одинаковые значения параметров задают группы
- Каждый следующий параметр задает группировку внутри групп предыдущего параметра

Основы SQL: выборка

```
SELECT * FROM `table_name`
   GROUP BY `num`;
+------+------+
| num  | id   |
+------+------+
|    1 |    1 |
|    2 |    1 |
|    3 |    2 |
+------+------+
```

Основы SQL: выборка

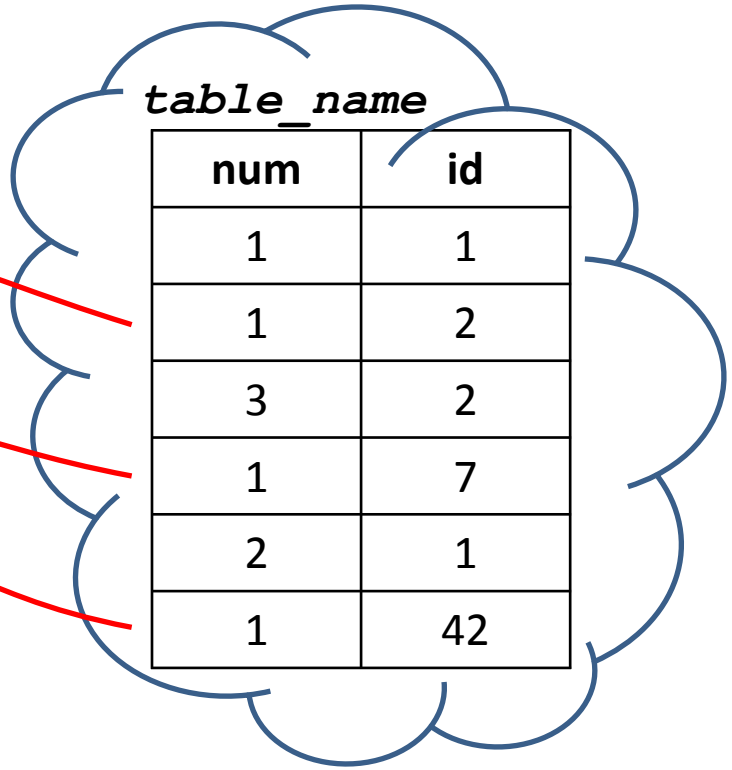
HAVING

- Аналог WHERE для ORDER BY
- Отбирает группы в выдачу
- Дописывается после ORDER BY

Основы SQL: выборка

```
SELECT * FROM table_name
  GROUP BY id HAVING id > 1;
```

```
+------+------+
| num  | id   |
+------+------+
|    1 |    2 |
|    1 |    7 |
|    1 |   42 |
+------+------+
```



Основы SQL: функции

SUM – суммирование для множества

```
SELECT SUM(num), SUM(id) FROM
  table_name;
```

```
+--------------+------------+
| SUM(`num`) | SUM(`id`) |
+--------------+------------+
|            9 |         55 |
+--------------+------------+
```

Основы SQL: функции

COUNT – подсчет строк в множестве

```
SELECT COUNT(`num`), COUNT(DISTINCT
  `num`) FROM `table_name`;
```

```
+--------------+-------------------------+
| COUNT(`num`) | COUNT(DISTINCT `num`)   |
+--------------+-------------------------+
|            6 |                       3 |
+--------------+-------------------------+
```

Основы SQL: функции

MIN, MAX – минимум/максимум

```
SELECT MIN(num), MAX(num)
FROM table_name;
```

```
+--------------+--------------+
|   MIN(num)   |   MAX(num)   |
+--------------+--------------+
|            1 |            3 |
+--------------+--------------+
```

Основы SQL: функции

AVG – среднее значение в множестве

```
SELECT AVG(num)
   FROM table_name;
+--------------+
|   AVG(num)   |
+--------------+
|       1.5000 |
+--------------+
```

Итоговые запросы (по всей выборке)

SUM (поле)-вычисляет суммы всех значений заданного поля или выражения в таблице или в каждой группе записей.

AVG(поле)-вычисляет среднее арифметическое заданного поля или выражения для всей таблице или для каждой группы.

MIN(поле), MAX(поле) - находят наименьшее и наибольшее значе-ния заданного поля или выражения в таблице или в каждой группе.

COUNT(поле) или COUNT(*)-находит число записей в таблице или в каждой группе.

FIRST(поле), LAST(поле)- находят первое и последнее значения заданного поля или выражения в таблице или в каждой группе.

Итоговые функции в групповых запросах

```
Select Kdf, Nazf,Count(*) As Kolvo,
  Max(God) As MaxGod, Min(God) As
  Mingod, Cint (Abs(Avg(Z1)-Avg(Z4)))As
  Rcen
From Firm
Where Z1>100
Group By Kdf, Nazf
Having Abs(Avg(Z1)-Avg(Z4))>10
Order By Abs(Avg(Z1)-Avg(Z4));
```

Статистические функции по подмножеству

DAvg - подсчет среднего арифметического значения столбца или выражения,

DCount - подсчет количества записей,

DFirst - нахождение первого значения столбца из группы,

DLast - нахождение последнего значения столбца из группы,

DMax - определение максимального значения столбца или выражения,

DMin - определение минимального значения столбца или выражения,

DSum - подсчет суммы значений столбца или выражения.

Синтаксис операторов следующий:

<имя_функции> ("выражение";"источник";"критерий")

Статистические функции по подмножеству

```
Select Kdf, Nazf, Izd, Kdiz, God
From Firm
Where God = DMax ("God", "Firm",
  "Kdf <>40") And Kdf<>40;
```

Прочтём «своими словами»:

Выбрать *Kdf*, *Nazf*, *Izd*, *Kdiz*, *God*
из *Firm* записи, для которых
Kdf отлично от 40, а также
God равно максимальному значению
God из Firm при Kdf не равном 40;