Синтаксис SQL

> Используемая СУБД – MySQL

```
cd "C:\Program Files\MySQL\MySQL Server 5.7\bin"

// Дамп базы данных
mysqldump -u имя_пользователя -p имя_базы > C:\dump.sql --single-transaction

// Если нужно сделать дамп только одной или нескольких таблиц
mysqldump -u имя_пользователя -p имя_базы имя_таблицы1 имя_таблицы2 > C:\dump.sql

// Заливаем бекап в базу данных
mysql -u имя_пользователя -p имя_базы < C:\dump.sql
```

Создание базы данных

После авторизации можно либо выбрать, либо создать новую базу данных:

```
USE DATABASE name_database;
CREATE DATABASE name_database;
CREATE DATABASE name_database CHARACTER SET utf8;

# Изменение кодировки для бд
ALTER DATABASE name_database CHARACTER SET utf8 COLLATE utf8_general_ci;

# Список всех баз данных данного пользователя
SHOW DATABASES;
```

Типы данных

Символьные

Тип	Значение	Диапазон
CHAR(20)	Символьная строка фиксированной длины. Незанятые символы – пробелы	до 255 символов
VARCHAR(20)	Символьная строка переменной длины	до 255 символов
TINYTEXT	Фиксированная длина (пробелы на конце)	255 символов
TEXT	Фиксированная длина (пробелы на конце)	65535 символов
MEDIUMTEXT	Фиксированная длина (пробелы на конце)	16 777 215 символов
LONGTEXT	Фиксированная длина (пробелы на конце)	4 294 967 295 символов

Столбец с другой кодировкой: `VARCHAR(20) character set utf8`

Числовые

Тип	Диапазон	Размер
BOOLEAN	[0; 1]	
TINYINT	[-128; 127], [0; 255]	2^8; 1 байт
SMALLINT	[-32768; 32767], [0; 65535]	2^16; 2 байта
MEDIUMINT	[-8 388 608; 8 388 606], [0; 16 777 2015]	2^24; 3 байта
INT	[-2 147 483 648; 2 147 483 647], [0; 4 294 967 295]	2^32; 4 байта
BIGINT	[-9 223 372 036 854 775 808; 9 223 372 036 854 775 807]	2^64; 8 байт
FLOAT(p, s)		
DOUBLE(p, s)		

Тип столбца: `smallint unsigned`

Временные

Тип	Формат	Диапазон
DATE	YYYY-MM-DD	от 1000-01-01 до 9999-12-31
DATETIME	YYYY-MM-DD HH:MI:SS	от 00:00:00 до 23:59:59
TIMESTAMP	YYYY-MM-DD HH:MI:SS	от 1970-01-01 до 2037-12-31
YEAR	YYYY	от 1901 до 2155
TIME	HHH:MI:SS	от -838:59:59 до 838:59:59

Построение таблицы

```
SHOW TABLES;    # список всех таблиц
DESC name_table;    # описание таблицы
DROP name_table;    # удаление таблицы
```

```
CREATE TABLE person (перечисление столбцов и ограничений)

# Столбцы
person_id SMALLINT UNSIGNED AUTO_INCREMENT, # автоматическое +1
address VARCHAR(30) NOT NULL,   # обязательное для заполнения поле
gender ENUM('M', 'F'),  # тип данных перечисление (только в MySql)

# Ограничения

# ограничение под именем pk_person на поле person_id (первичный ключ = уникальность значения,
только один на таблицу)
CONSTRAINT pk_person PRIMARY KEY (person_id)
PRIMARY KEY(person_id) # только для MySql
# комбинация столбцлв person_id и address должна быть уникальна
CONSTRAINT pk_person PRIMARY KEY (person_id, address)

# ограничение, не позволяет добавить в данную таблицу в столбец other_id значение,
# которого нет в таблице person в столбце person_id
CONSTRAINT fk_person FOREIGN KEY (other_id) REFERENCES person (person_id)
FOREIGN KEY (имя_колонки) REFERENCES сторонняя_таблица(имя_колонки)

// название ограничения -> имя таблицы -> столбец
CREATE UNIQUE INDEX subject_name_uindex ON subject (name);
```

Изменение столбцов

```
ALTER TABLE name_table MODIFY name_column SMALLINT UNSIGNED AUTO_INCREMENT;

ALTER TABLE name_table ADD CONSTRAINT pk_person PRIMARY KEY (ID, LastName);
ALTER TABLE persons ADD PRIMARY KEY (ID);
ALTER TABLE persons ADD FOREIGN KEY (имя_колонки) REFERENCES сторонняя_таблица(имя_колонки)

ALTER TABLE name_table DROP CONSTRAINT pk_person;
ALTER TABLE name_table DROP PRIMARY KEY;
```

Редактирование базы данных

insert

Вставка новой строки. Не обязательно предоставлять данные для всех столбцов, если только они не `not null`

```
INSERT INTO person
(person_id, name, gender, birth_date)   # список столбцов
VALUES (null, 'Artem', 'M', '1972-05-27');  # значения должны соответствовать столбцам
# null так как значение auto_increment
```

update

Изменение уже существующих строк

```
UPDATE person
SET address = '1225 Tremont St.',
    city = 'Boston',
    state = 'MA',
WHERE person_id = 1;
```

Если упустить блок where, то такие значения присвоились всем строкам таблицы

delete

Удаление данных из строки

```
DELETE FROM person
WHERE person_id = 2;
# WHERE person_id > 2;
```

Если упустить блок where, то из таблицы удалятся все данные

Извлечение информации из базы данных

Порядок следования операторов

Порядок выполнения запроса

- FROM
- ON
- JOIN
- WHERE
- GROUP BY
- WITH CUBE или WITH ROLLUP
- HAVING
- SELECT
- DISTINCT
- ORDER BY
- LIMIT

select

Выбор стобцов из таблицы

```
# Все столбцы из таблицы department
SELECT *
FROM department;

# Выбор двух столбцов
SELECT id, name
FROM department;
```

Также в запрос можно включать константы, выражения, встроенные функции

```
SELECT emp_id,
    'ACTIVE',  # в ответ добавиться колонка со значением ACTIVE
    emp_id * 3.14159,   # все значения этого столбца умножатся
    UPPER(lname)    # заглавные
FROM employee;
```

Можно добавить псевдонимы столбцов

```
SELECT emp_id,
    'ACTIVE' statusA,  # у колонки будет имя statusA
    emp_id * 3.14159 multipledByPi,
    UPPER(lname)
FROM employee;
```

Уничтожение дубликатов

```
SELECT DISTINCT cust_id
FROM account;
```

from

В этом блоке могут содержаться:

- Постоянные табицы,
- Временные таблицы (строки, возвращаемые подзапросом)
- Виртуальные таблицы (представления create view)

```
# Подзапросы
SELECT e.id, e.name
FROM (SELECT id, name, gender FROM employee) e;

# Представления - шаблонный запрос
CREATE VIEW employee_vw AS
SELECT id, name,
    YEAR(start_date) start_year
FROM employee;

SELECT id, start_year
FROM employee_vw;
```

Связи таблиц

Если в блоке присутствует более 1 таблицы, то их необходимо связать

```
# Соединение таблиц по id
SELECT employee.emp_id, employee.name, department.name dept_name
FROM employee INNER JOIN department
    ON employee.dept_id = department.dept_id;

# Псевдонимы таблиц
SELECT e.emp_id, e.name, d.name dept_name
FROM employee e INNER JOIN department d
    ON e.dept_id = d.dept_id;
```

where

Отбор данных

```
SELECT id, title
FROM books
WHERE title = 'Block';
```

Множественные условия (and, or, not). Разрешаются скобки

```
SELECT id, title
FROM books
WHERE title = 'Block'
    AND publish_date > '2012-01-01';
```

Условия вхождения в диапазон

```
# between
WHERE start_date BETWEEN '2001-01-01' AND '2003-01-01';

# in
WHERE product IN ('CHK', 'SAV', 'CD');
WHERE product NOT IN ('CHK', 'SAV', 'CD');
WHERE product_id IN (SELECT product_id FROM product
                     WHERE product_type_cd = 'ACCOUNT');
```

Символы маски

`_` – точно один символ `%` – либое число символов (в том числе ни одного)

```
WHERE name LIKE '_a%e%';
WHERE number LIKE '___-__-____';
WHERE name LIKE '_a%e%' OR name LIKE'a%';
```

Регулярные выражения

```
WHERE name REGEXP '^[FG]';
```

Работа с `null`

- Два `null` никогда не могут быть равны друг другу
- Значение равно 0, но это не `null`

```
# Только так проверять
WHERE id IS NULL;
```

group by

Сворачивание данных по определенному признаку, по группам

```
SELECT open_emp_id
FROM account
GROUP BY open_emt_id;
```

Агрегатные функции

Функции, которые применяются только к группированным данным, в частности к каждой группе

```
# Подсчет, сколько элементов в каждой группе
SELECT open_emp_id, COUNT(*) how_many
FROM account
GROUP BY open_emt_id;
#HAVING COUNT(*) > 4;
```

```
# Информация по каждому виду товару
SELECT product_cd,
    MAX(avail_balance) max_balance,
    MIN(avail_balance) min_balance,
    AVG(avail_balance) avg_balance,
    SUM(avail_balance) tot_balance,
    COUNT(*) num_accts
FROM account
GROUP BY product_cd;

# Подсчет значений в столбце
SELECT COUNT(open_emp_id)
FROM account;
#SELECT COUNT(DISTINCT open_emp_id) # уникальных
```

Группировка по нескольким столбцам

Агрегатные функции применяются к каждой подгруппе

```
SELECT product_cd, open_branch_id,
    SUM(avail_balance) tot_balance
FROM account
GROUP BY product_cd, open_branch_id;

# BUS   2   9354
# BUS   4   500
# CD    1   11500
# CD    4   0
```

Формирование обобщений

Сводная информация по каждой группе

```
SELECT product_cd, open_branch_id,
    SUM(avail_balance) tot_balance
FROM account
GROUP BY product_cd, open_branch_id WITH ROLLUP;

# BUS   2       9354
# BUS   4       500
# BUS   NULL    9854
# CD    1       11500
# CD    4       0
# CD    NULL    11500
```

having

Информация внутри групп фильтруется про помощи `having`, так как `where` выполняется еще до формирования групп

```
SELECT product_cd, SUM(avail_balance) prod_balance
FROM account
GROUP BY product_cd
HAVING MIN(avail_balance) >= 1000 AND MAX(avail_balance) <= 10000;
```

order by

Упорядочивание строк из результирующего набора

```
SELECT number, name
FROM account
ORDER BY number;    # по возрастанию
# ORDER BY number DESC;  # по убыванию
# ORDER BY RIGHT(id, 3);    # по выражению

# Упорядочивание по двум параметрам
SELECT number, name
FROM account
ORDER BY number, name;
# ORDER BY 1, 2;    # числовые заместители
```

Соединения таблиц

Декартово произведение

Будут получены все возможные комбинации строк из первой таблицы * строки второй таблицы

```
# Упущен блок ON
SELECT e.fname, e.lname, d.name
FROM employee e JOIN department d;

FROM employee e CROSS JOIN department d;
```

Внутреннее соединение

Пересечение двух таблиц. Будут соединены только строки, в которых есть одинаковые поля

```
SELECT e.name, d.name
FROM emploee e INNER JOIN department d
    ON e.dept_id = d.id;
```

Соединение трех и более таблиц

Последовательное соединение таблиц. Как снежный ком, после каждого соединения добавляются новые столбцы

```
SELECT a.account_id, c.fed_id, e.fname, e.lname
FROM customer c INNER JOIN account a
        ON a.cust_id = c.cust_id
    INNER JOIN employee e
        ON a.open_emp_id = e.emp_id;
```

Подзапросы в качестве таблиц

```
SELECT a.open_date, e.assigned_branch_id
FROM account a INNER JOIN
    (SELECT emp_id, assigned_branch_id
     FROM employee) e
ON a.open_emp_id = e.emp_id;
```

Рекурсивные соединения

Соединение таблицы с самой собой

```
SELECT e.fname, e.lname, e_mgr.fname mgr_fname, e_mgr.lname mgr_lname
FROM employee e INNER JOIN employee e_mgr
    ON e.superior_emp_id = e_mgr.emp_id;
```

Подзапросы

Это обычный запрос, содержащийся в другом SQL выражении. Может использоваться во всех выражениях для работы с данными (select, from, where, having, order by, update, insert, delete). Содержится в круглых скобках и выполняется до содержащего выражения

Может возвращать :

1. Одну строку и один столбец (можно использовать в выражениях)
2. Несколько строк и один столбец (in, all, any)
3. Несколько строк и столбцов

in, all, any, exists

Сравнивать нужно с наборами, в которых не содержится NULL. Потому что любое сравнение с NULL дает unknown результат

```
# Выберутся все строки, в которых emp_id не содержится в подзапросе
WHERE emp_id NOT IN (подзапрос_2)

# Выберутся все строки, в которых emp_id больше всех значений в подзапросе
WHERE emp_id > ALL (подзапрос_2)

# Выберутся все строки, в которых emp_id равен хотя бы одному из значений в подзапросе
WHERE emp_id = ANY (подзапрос_2)

# Проверка, вернул ли подзапрос хотя бы одну строку. Определение связи между данными
WHERE EXISTS (SELECT 1 FROM ...)
WHERE NOT EXISTS (SELECT 1 FROM ...)
```

Связанные подзапросы

Это подзапросы, которые зависят от своих содержащих выражений. В отличие от несвязанного вопроса, который выполняется непосредственно перед выполнением содержащего выражения, связанный подзапрос выполняется по разу для каждой строки-кандидата (это строки, которые могут быть включены в окончательный результат)

```
# c.cust_id делает этот подзапрос связанным.
# Он будет выполнятся для каждой строки таблицы customer
SELECT c.cust_id, c.cust_type_cd, c.city
FROM customer c
WHERE 2 = (SELECT COUNT(*)
           FROM account a
           WHERE a.cust_id = c.cust_id);

# Удаление строк, не удовлетворяющим условию
DELETE FROM department
WHERE NOT EXISTS (SELECT 1
                  FROM employee
                  WHERE employee.dept_id = department.dept_id);
```

Несвязанный подзапрос выполняется один раз. Связанный – столько, сколько строк в таблице

Объединения таблиц

Тблицы объединяются сверху вниз. Для изменения порядка можно использовать скобки

union

Объединение строк. Названия столбцов берутся по первому `select`

```
# Объединение с исключением дубликатов
SELECT emp_id
FROM employee
WHERE assigned_branch_id = 2 AND (title = 'Teller' OR title = 'Head Teller')                >
UNION
SELECT DISTINCT open_emp_id
FROM account
WHERE open_branch_id = 2;

# Объединение без исключения
UNION ALL
```

intersect

Пересечение таблиц (MySQL не позволяет)

```
...
INTERSECT
...
```

except

Возвращает первую таблицу за вычетом всех перекрытий со второй таблицей (MySQL не позволяет)

```
...
EXCEPT
...
```

Стандартные функции

Строковые

Числовые