Урок 1-2

Одна из основных функций SQL — получение данных из СУБД. Для построения всевозможных запросов к базе данных используется оператор **SELECT**. Он позволяет выполнять сложные проверки и обработку данных.

Ситуация, когда требуется сделать выборку по определенному условию, встречается очень часто. Для этого в операторе SELECT существует параметр **WHERE**, после которого следует условие для ограничения строк. Если запись удовлетворяет этому условию, то попадает в результат, иначе отбрасывается.

Общая структура запроса с оператором WHERE

```
SELECT поля_таблиц FROM список_таблиц
WHERE условия_на_ограничения_строк
[логический_оператор другое_условия_на_ограничения_строк];
```

Операторы сравнения

Операторы сравнения служат для сравнения 2 выражений, их результатом может являться ИСТИНА (1), ЛОЖЬ (0) и NULL.

Результат сравнения с NULL является NULL. Исключением является оператор эквивалентности.

Оператор	Описание
`=`	Оператор равенство
`<=>`	Оператор эквивалентность Аналогичный оператору равенства, с одним лишь исключением: в отличие от него, оператор эквивалентности вернет ИСТИНУ при сравнении `NULL <=> NULL`
`<>` или `!=`	Оператор неравенство
`<`	Оператор меньше
`<=`	Оператор меньше или равно
`>`	Оператор больше
`>=`	Оператор больше или равно

Логические операторы

Логические операторы необходимы для связывания нескольких условий ограничения строк.

- Оператор NOT — меняет значение специального оператора на противоположный
- Оператор OR — общее значение выражения истинно, если хотя бы одно из них истинно
- Оператор AND — общее значение выражения истинно, если они оба истинны

Специальные операторы

1. IS [NOT] NULL — позволяет узнать равно ли проверяемое значение NULL.

Для примера выведем всех членов семьи, у которых статус в семье не равен NULL:

```
SELECT * FROM FamilyMembers
WHERE status IS NOT NULL;
```

2. [NOT] BETWEEN min AND max — позволяет узнать расположено ли проверяемое значение столбца в интервале между min и max.

Выведем все данные о покупках с ценой от 100 до 500 рублей из таблицы Payments:

```
SELECT * FROM Payments
WHERE unit_price BETWEEN 100 AND 500;
```

3. [NOT] IN — позволяет узнать входит ли проверяемое значение столбца в список определённых значений.

Выведем имена членов семьи, чей статус равен «father» или «mother»:

```
SELECT member_name FROM FamilyMembers
WHERE status IN ('father', 'mother');
```

4. [NOT] LIKE шаблон — позволяет узнать соответствует ли строка определённому шаблону.

Например, выведем всех людей с фамилией «Quincey»:

```
SELECT member_name FROM FamilyMembers
WHERE member_name LIKE '% Quincey';
```

Трафаретные символы

В шаблоне разрешается использовать два трафаретных символа:

- символ подчеркивания (_), который можно применять вместо любого единичного символа в проверяемом значении
- символ процента (%) заменяет последовательность любых символов (число символов в последовательности может быть от 0 и более) в проверяемом значении.

Шаблон	Описание
`never%`	Сопоставляется любым строкам, начинающимся на «never».
`%ing`	Сопоставляется любым строкам, заканчивающимся на «ing».
`_ing`	Сопоставляется строкам, имеющим длину 4 символа, при этом 3 последних обязательно должны быть «ing». Например, слова «sing» и «wing».

Простые типы данных

CHAR(size) - Строки фиксированной длиной (могут содержать буквы, цифры и специальные символы). Фиксированный размер указан в скобках. Можно записать до 255 символов

VARCHAR(size) - Может хранить не более 255 символов.

INT(M) или INTEGER(M) - Целое число. Может быть объявлено положительным с помощью ключевого слова UNSIGNED, тогда элементам столбца нельзя будет присвоить отрицательное значение. Необязательный параметр M - количество отводимых под число символов.

NUMBER/NUMERIC - числовые данные

Создание таблицы

Для создания таблицы используется оператор CREATE TABLE. Его базовый синтаксис имеет следующий вид:

```
CREATE TABLE имя_таблицы (
    колонка_1 тип_данных,
   колонка_2 тип_данных,
   ...
   колонка_n тип_данных,
);
```

Например, создадим таблицу пользователей.

```
CREATE TABLE Users (
   id INT,
   name VARCHAR(255),
   age INT
);
```

Удаление таблицы

Удаление таблицы производится при помощи оператора DROP TABLE.

```
DROP TABLE имя_таблицы;
```

Добавление данных, оператор INSERT

Для добавления новых записей в таблицу предназначен оператор INSERT. Рассмотрим его общую структуру.

Общая структура запроса с оператором INSERT:

```
INSERT INTO имя_таблицы (поле_таблицы, ...)
VALUES (значение_поля_таблицы, ...)
```

Значения можно вставлять перечислением с помощью слова values, перечислив их в круглых скобках через запятую. Добавить новые записей можно следующими способами:

```
INSERT INTO Goods (good_id, good_name, type)
VALUES (5, 'Table', 3);
```

OR

```
INSERT INTO Goods VALUES (5, 'Table', 3);
```

Ограничения при создании таблицы

Ограничения позволяют указать определенные правила для данных, которые будут вставляться в таблицу. В случае, если мы попытаемся вставить данные, которые этим правилам не соответствуют, произойдет ошибка.

NOT NULL - Значение в поле не может быть null
UNIQUE - Значение в поле должно быть уникальным
PRIMARY KEY - NOT NULL + UNIQUE
CHECK - Соответствует какому-либо условию

Пример указания ограничений

```
CREATE TABLE ARTIST
(
        NAME  VARCHAR(255)  NOT NULL,
        LASTNAME  VARCHAR(255)  NOT NULL,
        gender char(1) check (gender in ('f', 'm'))
);
```

Особенности значения Null

В SQL существует значение Null, которое обозначает отсутствие значения в поле. Не стоит путать значение Null со значением 0 и пустой строкой (""). Также, их нельзя сравнивать ранее изученными операторами.

Для сравнения используются только IS NULL или IS NOT NULL

Автозаполнение

AUTO_INCREMENT/autoincrement позволяет создавать уникальный номер автоматически, когда новая запись вставляется в таблицу. Часто это поле основного ключа, которое мы хотели бы создать автоматически каждый раз, когда будет вставлена новая запись.

Синтаксис для MySQL
Следующий оператор SQL определяет столбец «user_id» как поле первичного ключа с автоматическим приращением в таблице «users»:

```
CREATE TABLE users (
   user_id int NOT NULL AUTO_INCREMENT,
   name varchar(255) NOT NULL,
   fullname varchar(255),
   balance int
);
```

MySQL использует ключевое слово AUTO_INCREMENT для выполнения функции автоматического увеличения.

По умолчанию начальное значение для AUTO_INCREMENT равно 1, и оно будет увеличиваться на 1 для каждой новой записи.

Чтобы вставить новую запись в таблицу «user», нам не нужно указывать значение для столбца «user_id», так как уникальное значение будет добавляться автоматически:

```
INSERT INTO users (name, fullname)
VALUES ('Том','Эдисон');
```