Тема 3. Основы работы с базами данных в MySQL. Выборки из одной таблицы.

Цель занятия:

Ознакомиться с основными командами и операциями, используемыми при работе с базами данных в MySQL, включая создание баз данных, таблиц и пользователей, выполнение запросов на выборку данных, а также команды для добавления, обновления и удаления данных.

Учебные вопросы:

1. Типы данных и ограничения.
2. DML-запросы. SELECT.
3. Арифметические операции в SELECT-запросах.
4. Оператор WHERE и фильтрация по условиям.
5. Сортировка данных при помощи ORDER BY.
6. Оператор LIMIT.

1. Типы данных и ограничения в MySQL.

В MySQL существуют различные типы данных и ограничения, которые помогают определить и контролировать структуру данных в таблицах.

Типы данных в MySQL

Числовые типы данных:

- TINYINT: Целое число от -128 до 127 или от 0 до 255 (беззнаковое). Используйте TINYINT UNSIGNED, когда отрицательные значения не нужны
- SMALLINT: Целое число от -32,768 до 32,767 или от 0 до 65,535 (беззнаковое).
- MEDIUMINT: Целое число от -8,388,608 до 8,388,607 или от 0 до 16,777,215 (беззнаковое).
- INT (INTEGER): Целое число от -2,147,483,648 до 2,147,483,647 или от 0 до 4,294,967,295 (беззнаковое).

- BIGINT: Целое число от -9,223,372,036,854,775,808 до 9,223,372,036,854,775,807 или от 0 до 18,446,744,073,709,551,615 (беззнаковое).
- FLOAT: Число с плавающей запятой (менее точное).
- DOUBLE: Число с плавающей запятой (более точное).
- DECIMAL: Число с фиксированной запятой (точность и масштаб указываются явно).

Строковые типы данных:

- CHAR(n): Строка фиксированной длины n (до 255 символов).
- VARCHAR(n): Строка переменной длины n (до 65,535 символов).
- TEXT: Текстовая строка переменной длины до 65,535 символов.
- MEDIUMTEXT: Текстовая строка до 16,777,215 символов.
- LONGTEXT: Текстовая строка до 4,294,967,295 символов.

Типы данных для даты и времени:

- DATE: Дата в формате YYYY-MM-DD.
- TIME: Время в формате HH:MM:SS.
- DATETIME: Дата и время в формате YYYY-MM-DD HH:MM:SS.
- TIMESTAMP: Метка времени в формате YYYY-MM-DD HH:MM:SS, автоматически обновляется при изменении строки.
- YEAR: Год в формате YYYY.

DATETIME и TIMESTAMP — это два типа данных в MySQL, используемые для хранения даты и времени, но они имеют несколько ключевых отличий:

- DATETIME: Диапазон: от '1000-01-01 00:00:00' до '9999-12-31 23:59:59’. Не зависит от временной зоны.
- TIMESTAMP: Диапазон: от '1970-01-01 00:00:01' UTC до '2038-01-19 03:14:07' UTC. Хранит значения в формате UTC и автоматически конвертирует их в локальное время при извлечении.
- TIMESTAMP: Может автоматически обновляться при вставке или изменении записи, если указано с помощью атрибута DEFAULT CURRENT_TIMESTAMP или ON UPDATE CURRENT_TIMESTAMP.

Когда использовать?

Используйте DATETIME, если Вам нужно хранить даты вне диапазона TIMESTAMP, Временная зона не имеет значения.

Используйте TIMESTAMP, если Вам нужно учитывать временные зоны, и нужно автоматическое обновление времени при изменении записей.

Типы данных для хранения двоичных данных:

- BINARY(n): Двоичные данные фиксированной длины n.
- VARBINARY(n): Двоичные данные переменной длины n.
- BLOB: Бинарный объект переменной длины до 65,535 байт.
- MEDIUMBLOB: Бинарный объект до 16,777,215 байт.
- LONGBLOB: Бинарный объект до 4,294,967,295 байт.

BLOB (Binary Large Object):

Файлы, изображения, видео: BLOB используется для хранения бинарных данных, таких как изображения, видео, файлы. Размеры варьируются: TINYBLOB, BLOB, MEDIUMBLOB, LONGBLOB.

BINARY и VARBINARY:

Двоичные данные, хэши: BINARY хранит фиксированную длину данных, VARBINARY — переменную длину. Подходят для хранения хэшей, бинарных данных.

Ограничения (Constraints) в MySQL

Ограничения используются для обеспечения целостности данных и управления данными в таблицах.

PRIMARY KEY:

Определяет уникальный идентификатор строки в таблице.

Поле с этим ограничением не может содержать NULL и должно быть уникальным.

Пример:

```
CREATE TABLE Employees (
    EmployeeID INT PRIMARY KEY,
    FirstName VARCHAR(50),
    LastName VARCHAR(50)
);
```

FOREIGN KEY:

Устанавливает связь между двумя таблицами, гарантируя, что значение в одном столбце (или наборе столбцов) совпадает со значением в другой таблице.

Поддерживает ссылочную целостность данных.

Пример

```
CREATE TABLE Orders (
    OrderID INT PRIMARY KEY,
    CustomerID INT,
    FOREIGN KEY (CustomerID) REFERENCES Customers(CustomerID)
);
```

UNIQUE:

Обеспечивает уникальность значений в столбце или наборе столбцов.

Пример:

```
CREATE TABLE Users (
    UserID INT PRIMARY KEY,
    Email VARCHAR(100) UNIQUE
);
```

NOT NULL:

Обязывает столбец содержать значение, то есть запрещает NULL.

Пример:

```sql
CREATE TABLE Products (
    ProductID INT PRIMARY KEY,
    ProductName VARCHAR(100) NOT NULL
);
```

DEFAULT:

Задает значение по умолчанию для столбца, если при вставке данных значение не указано.

Пример:

```sql
CREATE TABLE Orders (
    OrderID INT PRIMARY KEY,
    OrderDate DATE DEFAULT CURRENT_DATE
);
```

CHECK:

Ограничивает значения, которые могут быть вставлены в столбец, по определенному условию.

Пример (в MySQL 8.0+):

```
CREATE TABLE Employees (
    EmployeeID INT PRIMARY KEY,
    Age INT CHECK (Age >= 18)
);
```

Типы данных в MySQL помогают определить, какие именно данные могут быть сохранены в каждой колонке таблицы, а ограничения позволяют контролировать и защищать данные, обеспечивая целостность и уникальность.

Используя комбинацию различных типов данных и ограничений, можно создавать мощные и гибкие структуры баз данных, соответствующие различным требованиям приложений.

2. DML-запросы. SELECT.

DML (Data Manipulation Language) — это подмножество SQL, которое включает команды для работы с данными в базах данных.

В MySQL DML-запросы используются для добавления, обновления, удаления и выборки данных из таблиц.

Далее рассмотрим основные команды DML и примеры их использования.

1. **SELECT** — Выборка данных из таблицы

Команда SELECT используется для выборки данных из одной или нескольких таблиц.

Синтаксис:

SELECT column1, column2, ... **FROM** table_name;

Пример:

```
SELECT FirstName, LastName, Age
FROM Employees
WHERE Department = 'HR';
```

Этот запрос возвращает имена, фамилии и возраст всех сотрудников, работающих в отделе HR.

Общий вид структуры SELECT-запроса:

```
SELECT [DISTINCT] столбцы
FROM таблица
[JOIN другая_таблица ON условие_связи]
[WHERE условие_фильтрации]
[GROUP BY столбец_или_столбцы]
[HAVING условие_для_группы]
[ORDER BY столбец_или_столбцы [ASC | DESC]]
[LIMIT количество_строк [OFFSET смещение]];
```

Таблица для примеров:

id	name	price	category
1	iPhone 15	499000	smartphones
2	MacBook Pro	899000	laptops
3	AirPods	99000	audio
4	iPad	650000	tablets

Примеры:

Выбрать ВСЕ поля из таблицы products

SELECT * FROM products;

```
mysql> SELECT * FROM products;
+----+-------------+--------+-------------+
| id | name        | price  | category    |
+----+-------------+--------+-------------+
|  1 | iPhone 15   | 499000 | smartphones |
|  2 | MacBook Pro | 899000 | laptops     |
|  3 | AirPods     |  99000 | audio       |
|  4 | iPad        | 650000 | tablets     |
+----+-------------+--------+-------------+
```

Выбрать только имя и цену:

SELECT name, price FROM products;

```
mysql> SELECT name, price FROM products;
+--------------+--------+
| name         | price  |
+--------------+--------+
| iPhone 15    | 499000 |
| MacBook Pro | 899000 |
| AirPods      |  99000 |
| iPad         | 650000 |
+--------------+--------+
```

2. INSERT — Вставка данных в таблицу

Команда INSERT используется для добавления новых строк в таблицу.

Пример:

```
INSERT INTO Employees (FirstName, LastName, Age, Department)
VALUES ('John', 'Doe', 30, 'HR');
```

INSERT INTO Employees: Указывает, что вы добавляете новую запись в таблицу Employees.

(FirstName, LastName, Age, Department): Это список столбцов, в которые будут добавлены значения.

VALUES ('John', 'Doe', 30, 'HR'): Указывает значения, которые будут вставлены в соответствующие столбцы.

3. **UPDATE** — Обновление данных в таблице

Команда UPDATE используется для изменения существующих данных в таблице.

Пример:

```sql
UPDATE Employees
SET Age = 31
WHERE FirstName = 'John' AND LastName = 'Doe';
```

Этот запрос обновляет возраст сотрудника с именем John Doe на 31 год.

4. DELETE — Удаление данных из таблицы

Команда DELETE используется для удаления строк из таблицы.

Пример:

```
DELETE FROM Employees
WHERE LastName = 'Doe';
```

Этот запрос удаляет всех сотрудников с фамилией Doe из таблицы Employees.

Компоненты запроса:

SELECT: Обязательная часть запроса, указывающая, какие столбцы или выражения нужно выбрать.

Опционально можно использовать ключевое слово DISTINCT, чтобы удалить дубликаты из результата.

FROM: Обязательная часть, указывающая таблицу (или таблицы), из которых нужно выбирать данные.

JOIN: Опциональная часть, позволяющая объединять данные из нескольких таблиц на основе заданного условия.

WHERE: Опциональная часть, задающая условия фильтрации строк, которые должны быть включены в результат.

GROUP BY: Опциональная часть, группирующая строки, имеющие одинаковые значения в указанных столбцах.

Обычно используется с агрегатными функциями (например, COUNT, SUM, AVG).

HAVING: Опциональная часть, задающая условие фильтрации для групп, образованных с помощью GROUP BY.

ORDER BY: Опциональная часть, определяющая порядок сортировки результирующих строк.

Можно указать сортировку по возрастанию (ASC, по умолчанию) или по убыванию (DESC).

LIMIT: Опциональная часть, ограничивающая количество возвращаемых строк.

Параметр OFFSET позволяет пропустить указанное количество строк перед выборкой.

Пример полного SELECT-запроса:

```sql
SELECT DISTINCT FirstName, LastName, COUNT(*) AS NumOrders
FROM Customers
INNER JOIN Orders ON Customers.CustomerID = Orders.CustomerID
WHERE Orders.OrderDate >= '2024-01-01'
GROUP BY FirstName, LastName
HAVING COUNT(*) > 5
ORDER BY NumOrders DESC
LIMIT 10 OFFSET 5;
```

Этот запрос выбирает уникальные имена и фамилии клиентов, которые сделали более 5 заказов с 1 января 2024 года, сортирует результаты по количеству заказов в убывающем порядке и возвращает 10 строк, начиная с 6-й строки.

4. Арифметические операции в SELECT-запросах.

в SELECT-запросах можно использовать различные арифметические операторы для данных, хранящихся в таблицах:

- + – сложение;
- - – вычитание;
- / – деление;
- * – умножение;
- % – взятие остатка от деления.

Пример:

```
SELECT FirstName, Salary, Salary * 12 AS AnnualSalary
FROM Employees;
```

Этот запрос выбирает имя и зарплату сотрудника, а также вычисляет годовую зарплату, умножая месячную зарплату на 12.

```
SELECT FirstName, Salary, Salary + 1000 AS IncreasedSalary
FROM Employees;
```

Этот запрос выбирает имя и зарплату сотрудника, а также вычисляет зарплату с увеличением на 1000 единиц.

5. Оператор WHERE и фильтрация по условиям.

Оператор WHERE в SQL используется для фильтрации строк в результате запроса, чтобы вернуть только те записи, которые соответствуют заданным условиям.

Он позволяет ограничить выборку данных на основе различных критериев.

В качестве условий можно использовать:

- сравнения (=, >, <, >=, <=, !=);
- оператор IN;
- оператор BETWEEN;
- оператор LIKE.

С условиями можно применять логические операторы and, or и not:

- Оператор AND отображает запись, если оба операнда истинны;
- Оператор OR отображает запись, если хотя бы один операнд истинен;
- Оператор NOT инвертирует исходный операнд.

Примеры:

```
SELECT * FROM Employees
WHERE Department = 'HR';
```

Этот запрос выбирает все строки из таблицы Employees, где столбец Department равен 'HR’.

```
SELECT * FROM Employees
WHERE Age > 30 AND Department = 'HR';
```

Этот запрос выбирает все строки, где возраст больше 30 и отдел равен 'HR'.

```sql
SELECT title, release_year
FROM film
WHERE release_year >= 2000;
```

Этот запрос выбирает данные из таблицы film, отображая названия и года выпуска фильмов, которые были выпущены в 2000 году или позже.

```sql
SELECT first_name, last_name, active
FROM staff
WHERE NOT active = true;
```

Этот запрос выбирает данные из таблицы staff, отображая имена, фамилии и статус активности сотрудников, которые не активны.

Оператор **IN** в SQL используется для проверки, находится ли значение в заданном списке значений.

Он позволяет упростить запросы, которые требуют проверки на соответствие нескольким возможным значениям.

Синтаксис оператора IN

```
SELECT столбцы
FROM таблица
WHERE столбец IN (значение1, значение2, ..., значениеN);
```

Пример:

```sql
SELECT first_name, last_name, department
FROM staff
WHERE department IN ('HR', 'IT', 'Finance');
```

Этот запрос выбирает имена, фамилии и отделы сотрудников, которые работают в отделах HR, IT или Finance

Использование с подзапросом :

```sql
SELECT title, release_year
FROM film
WHERE film_id IN (
    SELECT film_id
    FROM rentals
    WHERE rental_date >= '2024-01-01'
);
```

Этот запрос выбирает названия и года выпуска фильмов, которые были арендованы начиная с 1 января 2024 года. Подзапрос возвращает идентификаторы фильмов, соответствующих условию.

Использование с отрицанием:

```sql
SELECT first_name, last_name
FROM staff
WHERE department NOT IN ('HR', 'IT');
```

Этот запрос выбирает имена и фамилии сотрудников, которые не работают в отделах HR или IT.

Оператор **BETWEEN** в SQL используется для фильтрации данных по диапазону значений.

Он позволяет выбрать строки, значения в которых находятся в заданном диапазоне, включая граничные значения.

Этот оператор может применяться как к числовым, так и к дата-временным данным.

Синтаксис оператора BETWEEN

```
SELECT столбцы
FROM таблица
WHERE столбец BETWEEN значение1 AND значение2;
```

Фильтрация по числовому диапазону:

```sql
SELECT first_name, salary
FROM employees
WHERE salary BETWEEN 30000 AND 50000;
```

Этот запрос выбирает имена и зарплаты сотрудников, чьи зарплаты находятся в диапазоне от 30,000 до 50,000 включительно.

Фильтрация по дате:

```sql
SELECT order_id, order_date
FROM orders
WHERE order_date BETWEEN '2024-01-01' AND '2024-12-31';
```

Этот запрос выбирает идентификаторы заказов и даты заказов, которые были сделаны в течение 2024 года, включая оба крайних дня.

Фильтрация с отрицанием:

```sql
SELECT first_name, last_name
FROM employees
WHERE salary NOT BETWEEN 30000 AND 50000;
```

Этот запрос выбирает имена и фамилии сотрудников, чьи зарплаты не находятся в диапазоне от 30,000 до 50,000.

Оператор LIKE в SQL используется для поиска строк, которые соответствуют определенному шаблону.

Это полезно для фильтрации данных, когда нужно найти строки, содержащие определенные подстроки или соответствующие определенным условиям.

Синтаксис оператора LIKE:

```
SELECT столбцы
FROM таблица
WHERE столбец LIKE шаблон;
```

Специальные символы шаблона в операторе LIKE:

%: Заменяет ноль или более символов.

_: Заменяет ровно один символ.

Поиск по начальной части строки:

```sql
SELECT first_name, last_name
FROM employees
WHERE first_name LIKE 'A%';
```

Этот запрос выбирает имена и фамилии сотрудников, чьи имена начинаются с буквы 'A'. Символ % заменяет любые символы после 'A'.

Поиск по содержимому строки:

```sql
SELECT email
FROM users
WHERE email LIKE '%@example.com';
```

Этот запрос выбирает электронные адреса пользователей, которые заканчиваются на '@example.com’.

Поиск по части строки с фиксированной длиной:

```sql
SELECT product_name
FROM products
WHERE product_code LIKE 'AB_123';
```

Этот запрос выбирает названия продуктов, где код продукта начинается с 'AB', за которым следует любой один символ (обозначенный _), затем '123'.

Использование NOT LIKE:

```sql
SELECT first_name, last_name
FROM employees
WHERE first_name NOT LIKE 'A%';
```

Этот запрос выбирает имена и фамилии сотрудников, чьи имена не начинаются с буквы 'A'.

Поиск по строкам, содержащим определенный набор символов:

```
SELECT address
FROM contacts
WHERE address LIKE '%Main St%';
```

Этот запрос выбирает данные из таблицы film, отображая названия и описания фильмов, в которых в поле description содержится слово "Scientist".

6. Сортировка данных при помощи ORDER BY.

Оператор ORDER BY в SQL используется для сортировки результатов запроса по одному или нескольким столбцам.

Он позволяет упорядочить строки в результирующем наборе данных в порядке возрастания или убывания.

Синтаксис оператора ORDER BY:

```
SELECT столбцы
FROM таблица
ORDER BY столбец1 [ASC|DESC], столбец2 [ASC|DESC], ...;
```

столбцы: Список столбцов, которые должны быть выбраны.

таблица: Имя таблицы, из которой будут извлечены данные.

столбец1, столбец2, ...: Столбцы, по которым будет производиться сортировка.

ASC: Сортировка по возрастанию (по умолчанию).

DESC: Сортировка по убыванию.

Сортировка по одному столбцу:

```sql
SELECT first_name, last_name
FROM employees
ORDER BY last_name ASC;
```

Этот запрос выбирает имена и фамилии сотрудников и сортирует их по фамилии в порядке возрастания (по умолчанию ASC).

Сортировка по нескольким столбцам:

```sql
SELECT first_name, last_name, hire_date
FROM employees
ORDER BY hire_date DESC, last_name ASC;
```

Этот запрос выбирает имена, фамилии и даты найма сотрудников. Сначала сортирует по дате найма в порядке убывания, а затем по фамилии в порядке возрастания для сотрудников, нанятых в один и тот же день.

Сортировка числовых значений:

```sql
SELECT product_name, price
FROM products
ORDER BY price DESC;
```

Этот запрос выбирает названия продуктов и их цены, сортируя их по цене в порядке убывания.

7. Оператор LIMIT.

Оператор LIMIT в SQL используется для ограничения числа строк, возвращаемых запросом. Это полезно для оптимизации запросов и управления объемом данных, возвращаемых клиенту. Особенно актуально это при работе с большими таблицами или при реализации постраничного отображения данных.

Синтаксис оператора LIMIT:

```
SELECT столбцы
FROM таблица
ORDER BY столбец
LIMIT количество_строк;
```

Возвращение первых N строк:

```sql
SELECT first_name, last_name
FROM employees
ORDER BY hire_date DESC
LIMIT 5;
```

Этот запрос выбирает имена и фамилии пяти самых последних нанятых сотрудников, упорядоченных по дате найма в порядке убывания.

Возвращение строк с определенного номера:

```sql
SELECT first_name, last_name
FROM employees
ORDER BY hire_date DESC
LIMIT 10 OFFSET 5;
```

Этот запрос выбирает 10 сотрудников, начиная с 6-го (пропустив первые 5), упорядоченных по дате найма в порядке убывания.

Контрольные вопросы:

- Что такое DDL-запросы? Приведите примеры основных команд DDL.
- Какие типы данных используются в MySQL для хранения текстовых и числовых значений? Приведите примеры.
- Объясните, чем отличаются команды CREATE, ALTER и DROP в MySQL.
- Какой оператор используется для фильтрации данных по условиям в запросе? Приведите пример запроса с использованием этого оператора.
- Чем отличается тип данных VARCHAR от TEXT в MySQL? В каких случаях их лучше использовать?
- Какие команды относятся к DML-запросам? Приведите примеры их использования.
- Что делает оператор ORDER BY в SQL? Как с его помощью можно отсортировать данные по убыванию?
- Для чего используется оператор LIMIT в SQL? Приведите пример его использования.
- Как можно объединить несколько условий фильтрации в запросе с помощью оператора WHERE? Приведите пример.
- Какие арифметические операции можно выполнять в SELECT-запросах? Приведите примеры.

Практика:

https://sql-academy.org/ru/guide