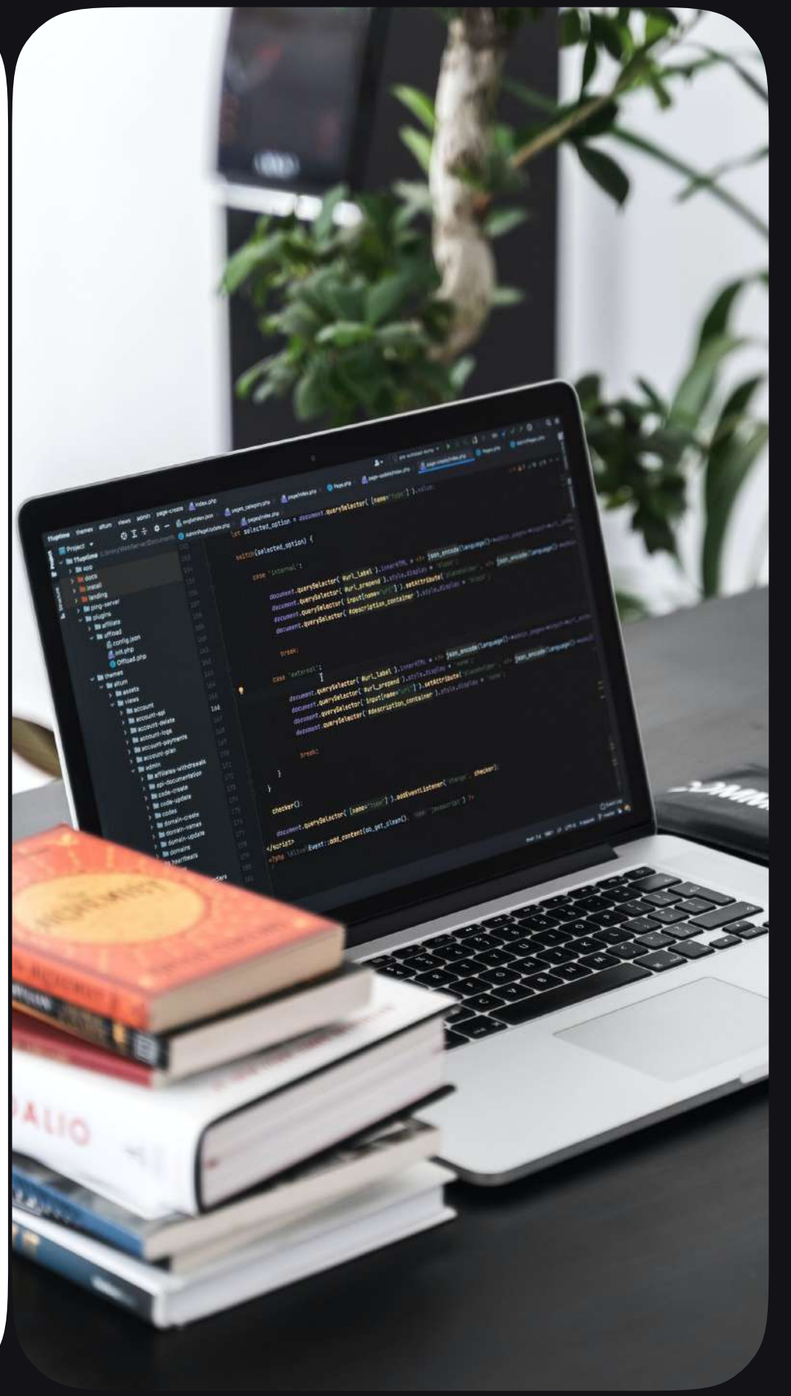
Занятие 10

Простые и составные запросы в SQL



Простые и составные SQL запросы

Условный оператор WHERE

Группировка и агрегирующие функции

Вложенные запросы

Многотабличные запросы

Добавление, удаление, обновление данных таблиц

Оператор SELECT

Одна из основных функций SQL — **получение** данных и **управление** данными в БД. Для получения данных и построения различных запросов к БД используется оператор **SELECT**.

Общая структура запроса с оператором **WHERE** выглядит так:

SELECT [**DISTINCT** | **ALL**] поля_таблиц

FROM список_таблиц

[**WHERE** условия_на_ограничения_строк]

[**GROUP BY** условия_группировки]

[**HAVING** условия_на_ограничения_строк_после_группировки]

[**ORDER BY** порядок_сортировки [**ASC** | **DESC**]]

[**LIMIT** ограничение_количества_записей];

SELECT

Для того, чтобы вывести все столбцы из таблицы можно выполнить запрос

SELECT * FROM users;

Если необходимо вывести только определенные столбцы таблицы

SELECT name, email FROM users;

Можно выполнить запрос и использовать псевдоним для названия столбца

SELECT name AS username FROM users;

Иногда необходимо получить только уникальные записи. Оператор **DISTINCT** служит для удаления повторяющихся записей. Например, выведем список профессий:

SELECT DISTINCT name FROM jobs;

WHERE

Чаще всего нужно получать
не все данные из таблиц БД,
а только удовлетворяющие
какому-либо условию.
Для этого существует
оператор **WHERE**.

Общая структура запроса
с оператором **WHERE** выглядит так:

SELECT поле_таблицы1, …

FROM таблица

WHERE условие_на_ограничения

В операторе **WHERE** применяются операторы сравнения,
специальные и логические операторы.

Операторы сравнения: **>**, **<**, **>=**, **<=**, **=**, **!=** (не равно)

WHERE

Специальные операторы

IS NULL

Проверяет является ли проверяемое значение **NULL** или IS NOT NULL проверяет что не является

IN

Проверяет входит ли проверяемое значение в список определённых значений. **NOT IN** проверяет что не входит

BETWEEN min AND max

Проверяет расположено ли проверяемое значение в интервале между min и max. **NOT BETWEEN** проверяет что не расположено

LIKE

Проверяет соответствует ли строка определённому шаблону. **NOT LIKE** проверяет что не соответсвтует

WHERE

Шаблон может содержать символ:

_ — один любой символ

% — любые символы

Логические операторы необходимы объединения нескольких условий:

NOT

меняет значение специального или логического оператора на противоположный

AND, OR

проверка нескольких условий с помощью операций «**И**» или «**ИЛИ**»

Агрегирующие функции

Для выполнения вычислений, например, поиска максимума или минимума, суммы или количества, существуют **агрегирующие** (или агрегатные) функции.

Выборка также может быть ограничена с помощью **WHERE**

COUNT()

возвращает количество строк в результирующей выборке

MIN() и MAX()

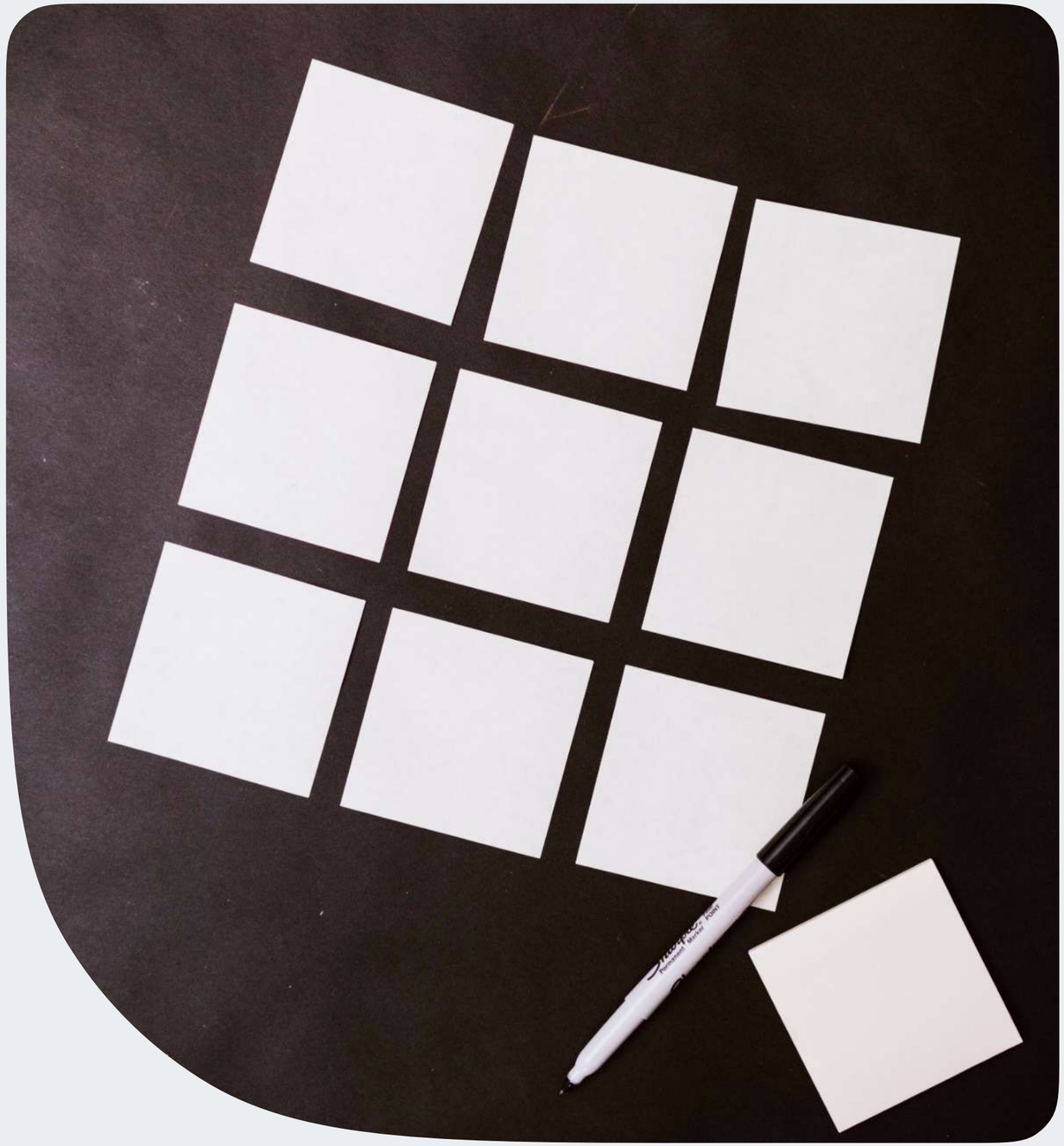
функции для поиска максимального или минимального значения в столбце

AVG() и SUM()

функции для поиска среднего и суммы в столбце

Группировка GROUP BY

Иногда требуется группировать результирующую выборку по одному или нескольким столбцам. Для этого используется оператор **GROUP BY** и агрегирующие функции. Для фильтрации строк после группировки используется оператор **HAVING**.



Группировка GROUP BY

Общая структура запроса с группировкой выглядит так:

SELECT [поля, агрегирующие_функции]

FROM имя_таблицы

GROUP BY поля_группировки

HAVING условие_на_ограничение_строк_после_группировки

Отличия **HAVING** от **WHERE**:

WHERE	**HAVING**
сначала выбираются записи по условию, а затем могут быть сгруппированы, отсортированы и т.д.	сначала группируются записи, а затем выбираются по условию, при этом, в отличие от WHERE, в нём можно использовать значения агрегатных функций

Вложенные запросы

Вложенный запрос или **подзапрос** — это запрос на выборку, который используется внутри других инструкции SELECT, INSERT, UPDATE или DELETE

```
SELECT AVG(salary)
FROM users
WHERE job_id in (
    SELECT id
    FROM jobs
    WHERE name = 'Doctor'
);
```

Например найдем среднюю зарплату пользователей с названием работы Doctor

Сортировка ORDER BY

Для того, чтобы отсортировать результат запроса в SQL есть оператор **ORDER BY**.

Общая структура запроса с группировкой выглядит так:

```
SELECT поля_таблиц
FROM таблица
ORDER BY столбец_1 [ASC|DESC]
```

В квадратных скобках указаны необязательные параметры, определяющие порядок сортировки.

- **ASC** — сортировка по возрастанию
- **DESC** — сортировка по убыванию

Если ничего не указывать, то по умолчанию выполняется сортировка по возрастанию.

Многотабличные запросы

Для объединения таблиц используется оператора JOIN.

Общая структура многотабличного запроса выглядит так:

SELECT названия_столбцов

FROM таблица_1

JOIN таблица_2 **ON** условие_соединения

Объединим две таблицы по условию совпадения полей job_id таблицы users и id таблицы jobs:

```
SELECT users.name, jobs.name

FROM users

JOIN jobs ON users.job_id = jobs.id;
```

Многотабличные запросы

Соединение JOIN бывает внутренним (INNER) или внешним (OUTER), левым (LEFT), правым (RIGHT) и полным (FULL).

При выполнении запросов с (**INNER**) **JOIN** в результирующую выборку попадут только те записи, в которых выполняется условие, заданное в **ON**.

При выполнении запросов с **LEFT** (**OUTER**) **JOIN** в выборку попадут все строки левой таблицы. Данными из правой таблицы дополняются те строки левой таблицы, для которых выполняются условие заданное в **ON**. Для недостающих данных вместо строк правой таблицы будут вставлены пустые значения (**NULL**).

RIGHT (**OUTER**) **JOIN** — работает также как и **LEFT JOIN**, но выводятся все записи из правой таблицы, а к ним добавляются строки из левой таблицы для которых выполняются условие заданное в **ON**. Для недостающих данных вместо строк левой таблицы будут вставлены пустые значения (**NULL**).

Многотабличные запросы

FULL (**OUTER**) **JOIN** выведет все записи из объединяемых таблиц. Записи, у которых выполняется условие **ON** — выводятся парами, у остальных будут вставлены пустые значения (**NULL**).

Добавление, удаление и обновление данных таблиц

Для добавления новых записей в таблицу предназначен оператор **INSERT**.

Общая структура запроса с оператором INSERT:

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

Добавим нового пользователя в таблицу users:

```
INSERT INTO users (name, job_id) VALUES('Michael Jordan', 1);
```

Добавление, удаление и обновление данных таблиц

Для обновления записей в таблице существует оператор **UPDATE**.

Общая структура запроса с оператором UPDATE:

UPDATE имя_таблицы

SET поле_таблицы1 = значение_поля_таблицы1, …

WHERE условие_выборки

Изменим данные нового пользователя:

```
UPDATE users
SET job_id = 2, salary = 50000
WHERE name = 'Michael Jordan';
```

Добавление, удаление и обновление данных таблиц

Для удаления записей используется оператор **DELETE**.

Общая структура запроса с оператором DELETE:

DELETE FROM имя_таблицы

WHERE условие_отбора_записей

Добавим нового пользователя в таблицу users:

```
DELETE FROM users
WHERE name = "Michael Jordan";
```