04 - SQL

Adam Terlo

15.04.2024

Материалы

github.com/adamxrvn/hse-lyceum-android-course

Введение в SQL

SQL (Structured Query Language) — это стандартизированный язык программирования, используемый для управления реляционными базами данных и выполнения различных операций с данными в них.

Что такое СУБД?

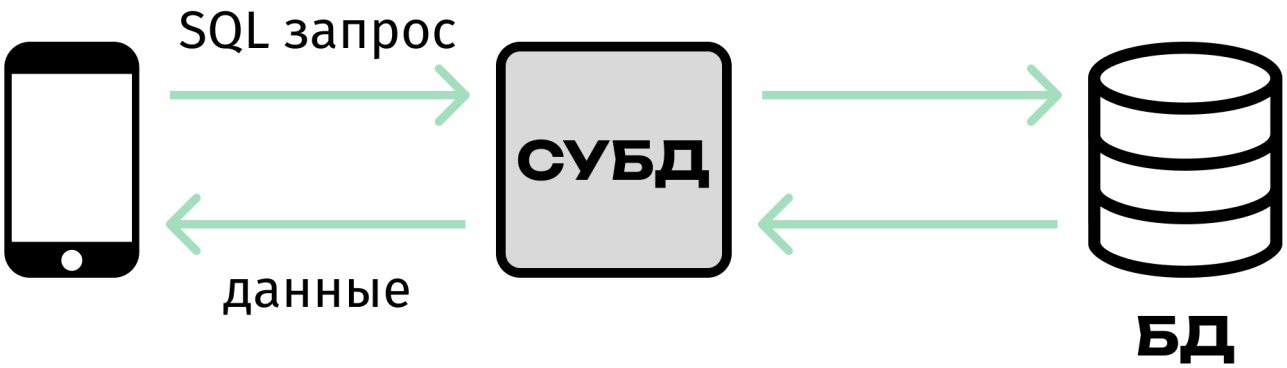

Рис 1. СУБД

Для работы с БД и SQL нам требуется система управления базами данных (СУБД) — ПО предназначенное для создания, управления и манипуляции данными в БД.

Типы СУБД

Реляционные СУБД (SQL)

- **SQLite**: Легковесная, встраиваемая СУБД, идеально подходит для мобильных приложений и небольших проектов.
- **MySQL**: Одна из самых популярных открытых СУБД, широко используется в веб-разработке.
- **PostgreSQL**: Мощная, расширяемая СУБД, часто используется для сложных и больших систем.

Нереляционные СУБД (NoSQL)

- **MongoDB**: Документо-ориентированная СУБД, оптимизирована для хранения и обработки больших объемов неструктурированных данных.
- **Cassandra**: Распределенная СУБД, предназначенная для обработки больших данных с высокой доступностью без единой точки отказа.

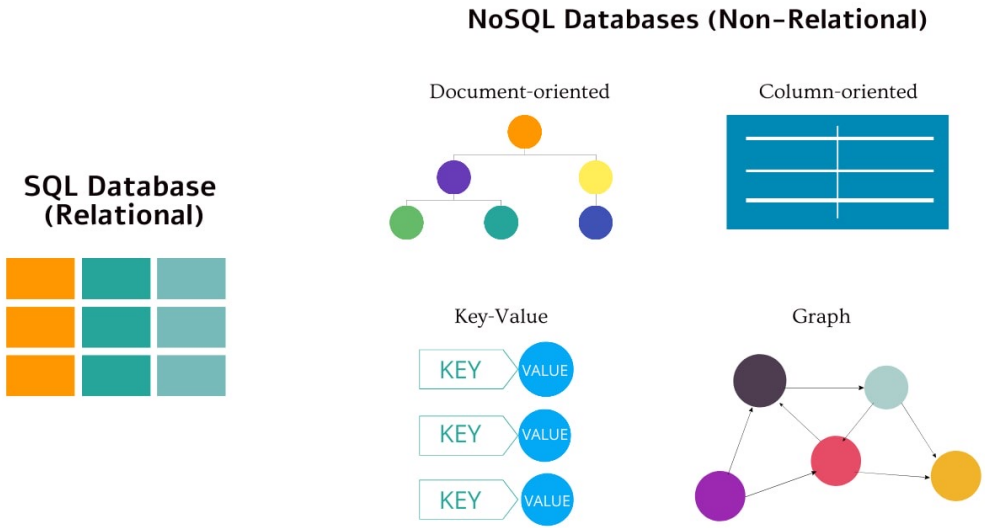

Рис 2. SQL vs NoSQL

Что такое реляционные базы данных?

Реляционные базы данных состоят из таблиц, которые содержат данные, организованные в строки и столбцы. Каждая таблица представляет собой набор связанных данных и включает:

- **Таблицы (Tables)**: Хранят информацию об объектах одного типа.
- **Столбцы (Columns)**: Определяют типы данных, например, имя, возраст, баланс счета.
- **Строки (Rows)**: Конкретные записи или данные.
- **Ключи (Keys)**: Помогают уникально идентифицировать строки в таблице, например, первичный ключ.

Ключи в реляционных БД

Ключи в реляционных базах данных — это специальные поля в таблицах, которые помогают уникально идентифицировать строки, а также устанавливать и поддерживать связи между различными таблицами.

Основные типы ключей

Primary Key

- **Описание**: Уникально идентифицирует каждую строку в таблице. В таблице может быть только один первичный ключ, который не может принимать NULL.
- **Пример**: В таблице клиентов, `Client_ID` может быть первичным ключом.

Основные типы ключей

Foreign Key

- **Описание**: Ссылается на первичный ключ другой таблицы, обеспечивая связь между таблицами. Позволяет поддерживать целостность данных между таблицами.
- **Пример**: В таблице счетов, `Client_ID` может быть внешним ключом, который ссылается на первичный ключ в таблице клиентов.

Основные типы ключей

Unique Key

- **Описание**: Гарантирует, что все значения в столбце или комбинации столбцов уникальны. Может принимать NULL, если только это явно не запрещено.
- **Пример**: Номер паспорта в таблице клиентов может быть уникальным ключом.

Типы данных

Числовые типы данных

Тип данных	Размер	Диапазон	Описание
INT	4 байта	-2,147,483,648 до 2,147,483,647	Целое число
SMALLINT	2 байта	-32,768 до 32,767	Маленькое целое число
BIGINT	8 байт	-9,223,372,036,854,775,808 до 9,223,372,036,854,775,807	Большое целое число
DECIMAL	Различно	Различно в зависимости от заданных параметров точности и масштаба	Число с фиксированной точностью
FLOAT	4 байта	-3.4E+38 до 3.4E+38	Одинарная точность с плавающей точкой
DOUBLE	8 байт	-1.7E+308 до 1.7E+308	Двойная точность с плавающей точкой

Текстовые типы данных

Тип данных	Размер	Описание
CHAR	1 байт на символ	Фиксированная длина, хранит N символов
VARCHAR	Различно	Переменная длина, хранит до N символов
TEXT	Не ограничен	Для больших текстов

Другие типы данных

Тип данных	Размер	Описание
BOOLEAN	1 байт	Хранит значение истина или ложь
DATE	3 байта	Хранит дату (год, месяц, день)
TIME	3-4 байта	Хранит время (час, минута, секунда)
DATETIME	8 байт	Хранит и дату, и время
TIMESTAMP	4 байта	Хранит время в формате Unix timestamp

Пример базы данных

Таблица: Clients

Client_ID (PK)	Name	Surname	Middle_Name	Date_Of_Birth	Address
1	Иван	Иванов	Иванович	1985-02-15	ул. Ленина, 10
2	Мария	Петрова	Сергеевна	1990-07-22	ул. Советская, 8
3	Михаил	Алексеев	Сергеевич	1972-09-05	Барвиха 52

Таблица: Accounts

Account_ID (PK)	Client_ID (FK)	Balance
101	1	15000
102	2	25000
103	3	192000000

Основные операции SQL

Создание таблиц

Пример таблицы `Clients`, которая будет хранить информацию о клиентах:

```
CREATE TABLE Clients (
    Client_ID INT PRIMARY KEY,
    Name VARCHAR(100),
    Surname VARCHAR(100),
    Middle_Name VARCHAR(100),
    Date_Of_Birth DATE,
    Address VARCHAR(255)
);
```

Основные операции SQL

Создание таблиц

Пример таблицы `Accounts`, которая будет содержать информацию о банковских счетах клиентов:

```
CREATE TABLE Accounts (
    Account_ID INT PRIMARY KEY,
    Client_ID INT,
    Balance DECIMAL(15, 2),
    FOREIGN KEY (Client_ID) REFERENCES
Clients(Client_ID)
);
```

Основные операции SQL

SELECT (Выборка данных)

Используется для чтения данных из одной или нескольких таблиц.

Пример: получение списка клиентов с балансом выше 10,000.

```
SELECT name, balance FROM accounts WHERE balance > 10000;
```

Основные операции SQL

INSERT (Вставка данных)

Позволяет вставлять новые строки в таблицу.

Пример: регистрация нового клиента в системе.

```
INSERT INTO clients (name, address) VALUES ('Иван Иванов', 'ул. Ленина, 10');
```

Основные операции SQL

UPDATE (Обновление данных)

Модифицирует существующие строки в таблице.

Пример: добавление суммы к балансу существующего клиента.

```
UPDATE accounts SET balance = balance + 500 WHERE
client_id = 101;
```

Основные операции SQL

DELETE (Удаление данных)

Удаляет строки из таблицы.

Пример: удаление информации о клиенте банка. Он нас покинул 💀

```
DELETE FROM clients WHERE client_id = 101;
```

Оператор ORDER BY

Оператор `ORDER BY` используется в SQL для сортировки результатов запроса по одному или нескольким столбцам, в порядке возрастания или убывания.

Пример: сортировка по одному столбцу

Сортировка списка клиентов по фамилии в алфавитном порядке:

```
SELECT Name, Surname FROM Клиенты
ORDER BY Surname ASC;
```

Оператор ORDER BY

Пример: сортировка по нескольким столбцам

Сортировка списка клиентов сначала по фамилии, затем по имени:

```
SELECT Name, Surname FROM Клиенты
ORDER BY Surname ASC, Name ASC;
```

`ASC` означает сортировку по возрастанию, `DESC` — по убыванию. Если направление не указано, по умолчанию используется `ASC`.

Оператор LIMIT

Оператор `LIMIT` применяется в SQL для ограничения количества строк, возвращаемых запросом. Это особенно полезно при работе с большими объемами данных или когда нужно получить только первые несколько записей результата.

Пример: ограничение количества строк

Получение первых пяти клиентов из таблицы:

```
SELECT Name, Surname FROM Клиенты
ORDER BY Surname
LIMIT 5;
```

Оператор LIMIT

Пример: ограничение с указанием смещения

Пропуск первых пяти записей и выбор следующих пяти:

```
SELECT Name, Surname FROM Клиенты
ORDER BY Surname
LIMIT 5 OFFSET 5;
```

`LIMIT` часто используется для пагинации результатов на веб-страницах или в приложениях, где необходимо показывать данные порциями.

Эти команды помогут контролировать объем данных, загружаемых из базы данных, что может существенно ускорить время загрузки и уменьшить нагрузку на сервер.

Агрегатные функции

Агрегатные функции используются для выполнения вычислений над набором значений и возвращают одиночное значение. Они часто применяются в аналитических запросах и отчетах.

Примеры основных агрегатных функций

AVG (Среднее значение)

Возвращает среднее значение числового столбца.

```
SELECT AVG(Balance) FROM accounts;
```

SUM (Сумма)

Возвращает сумму значений столбца.

```
SELECT SUM(Balance) FROM accounts WHERE
Account_Type = 'сберегательный';
```

Примеры основных агрегатных функций

COUNT (Количество)

Возвращает количество строк, удовлетворяющих критерию.

```
SELECT COUNT(*) FROM clients WHERE Status = 'VIP';
```

MAX (Максимальное значение)

Возвращает наибольшее значение в столбце.

```
SELECT MAX(Balance) FROM accounts;
```

Оператор JOIN

`JOIN` используется для объединения строк из двух или более таблиц, основываясь на общем столбце между ними.

Типы JOIN

INNER JOIN, LEFT JOIN, RIGHT JOIN, FULL OUTER JOIN, etc.

Типы JOIN

INNER JOIN

Возвращает строки, когда есть совпадение в обеих таблицах.

```
SELECT clients.Name, accounts.Account_Number FROM
clients
INNER JOIN accounts ON clients.Client_ID =
accounts.Client_ID;
```

Типы JOIN

LEFT JOIN

Возвращает все строки из левой таблицы и совпадающие строки из правой таблицы. Если совпадений нет, результат будет содержать `NULL` на месте столбцов правой таблицы.

```
SELECT clients.Name, accounts.Account_Number FROM
clients
LEFT JOIN accounts ON clients.Client_ID =
accounts.Client_ID;
```

Типы JOIN

RIGHT JOIN

Аналогичен `LEFT JOIN`, но возвращает все строки из правой таблицы.

```
SELECT clients.Name, accounts.Account_Number FROM
clients
RIGHT JOIN accounts ON clients.Client_ID =
accounts.Client_ID;
```

Типы JOIN

FULL OUTER JOIN

Возвращает строки, когда есть совпадение в одной из таблиц.

```
SELECT clients.Name, accounts.Account_Number FROM
clients
FULL OUTER JOIN accounts ON clients.Client_ID =
accounts.Client_ID;
```

Расширенные возможности SQL

SQL также поддерживает более сложные операции, такие как:

- **Транзакции**: Гарантируют безопасность операций путем их группировки. Если одна операция в группе транзакции не удается, то все остальные откатываются к исходному состоянию.
- **Триггеры**: Автоматические скрипты, которые выполняются при определенных событиях в базе данных.
- **Индексы**: Улучшают скорость поиска данных в таблицах.
- **Агрегатные функции**: Позволяют выполнять вычисления на наборе значений, например, сумма или среднее.
- **Оконные функции**: инструменты, которые позволяют выполнять сложные вычисления по наборам строк, которые связаны с текущей строкой.
- **Group By**: оператор, который используется для группировки строк в более крупные наборы на основе одного или нескольких столбцов.

Зачем нужен SQL в Android и Room?

В контексте Android, SQL используется для работы с локальной базой данных на устройстве. Room — это абстракция над SQLite, которая позволяет удобнее работать с базой данных с помощью объектно-ориентированного подхода, упрощая многие сложные аспекты работы с SQLite.

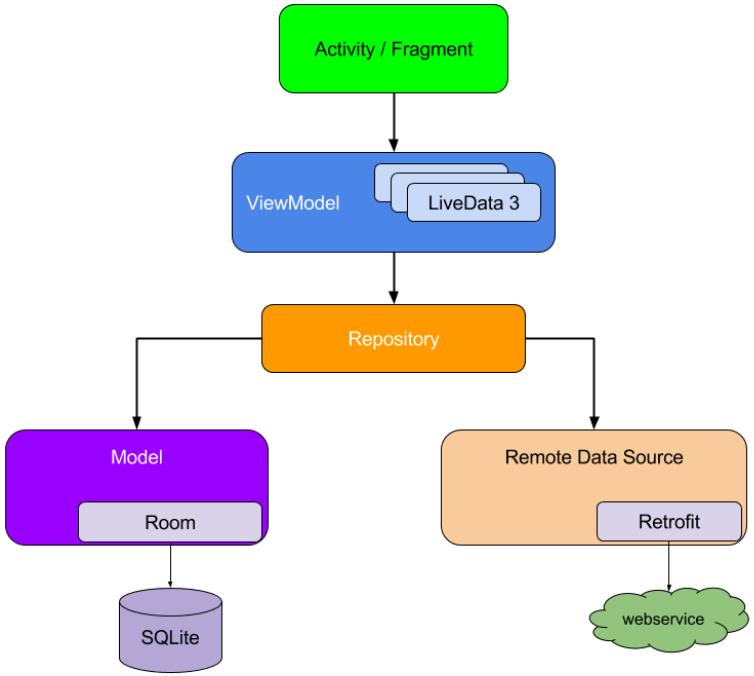

Рис 3. Архитектура MVVM с использованием Room и Retrofit. Автор?

Задачи

1. Напишите запрос для получения списка всех клиентов, включая их имя, фамилию и адрес. Отсортируйте результаты по фамилии в алфавитном порядке.
2. Получите список счетов с балансом выше 100,000, указав тип счета и баланс. Результаты должны быть отсортированы по убыванию баланса.
3. Напишите запрос для вывода имени клиента и номера счета для всех счетов с типом “сберегательный”.
4. Выполните запрос, который идентифицирует всех клиентов, имеющих отрицательный баланс хотя бы на одном из своих счетов. Выведите имена и фамилии должников, номер каждого счета с отрицательным балансом и сам баланс.
5. Также посчитайте общую сумму долга по всем счетам.
6. Обновите информацию о балансе для всех счетов типа “сберегательный”. Увеличьте баланс на 5000 для счетов, у которых текущий баланс меньше 10000.
7. Банк прекратил сейфовых пространств в “г. Урюпинск”, удалите все активные ячейки данного города из базы.
8. Клиент с Client_ID = 52 арендовал две ячейки в посёлке Дубай. Необходимо добавить данные об аренде в таблицу bank_lockers. Ячейки имеют номера “Locker-77777” и “Locker-00001”, сроки аренды составляют 6 и 12 месяцев соответственно.