SQLite-like DB

Романенко Никита
Образцов Дмитрий
Сидоренко Софья
Мункуев Владислав

НГУ, ФИТ 2025

Зачем мы это делаем?

Во многих приложениях не требуется полноценная серверная СУБД,но при этом возникает необходимость:

- Локально хранить структурированные данные
- Выполнять SQL-подобные запросы
- Делать простые аналитические выборки
- Не поднимать сервер

Наши плюсы

- Данные **хранятся напрямую в файловой системе** пользователя, соответственно это снижает расходы на запросы и упрощает использование.
- Из-за **колоночного хранения** в каких-то случаях ускоряется выборка данных. К примеру если таблица широкая(много колонок). Также при выборке столбцов будут считываться только отдельные файлы, что оптимизирует расход памяти

Наши цели

Создать свою SQLite-like БД которая поддерживает:

- Колоночное представление данных
- Простое журналирование
- Минимальный набор запросов
- Простую индексацию
- Бинарное хранение данных

Разработать драйвер JDBC для нашей БД

- Разработать не сетевой драйвер для подключения к нашей БД из java-программ

Системные требования

ОС:
macOS ver. 13+
Ubuntu ver. 24.04+
Windows 10+

Java:
Java ver. 17+

TUI:
Работает в текстовом режиме

Что готово на данный момент?

MVP БД обрабатывающая минимальный набор запросов:

DDL запросы: CREATE, DROP, ALTER, USE

DML запросы: SELECT, INSERT

Хранение данных организовано в формате JSON

Как происходит Парсинг?

Парсинг осуществляется с помощью ANTLR

- Лексер разбивает строку на токены(SELECT, CREATE, INTEGER, «строка», специальные символы и тд.
- Парсер по правилам строит Parse tree с контекстами(к примеру SelectDataStatementContext)
- Дальше по дереву проходится обработчик(visitor) и вытаскивает из контекстов параметры запроса(имя таблицы, выбранные колонки и тд.)

Обработка запроса

За обработку отвечает класс FileManager

- Каждая команда проходит через общий слой доступа к файловому хранилищу(FileManager).

Хранение данных

- Каждая база лежит в отдельной папке/директории.
- Каждая таблица имеет свои метаданные(список колонок, типы и тд.).
- Данные каждого столбца лежат в отдельном файле, т.е. у нас **колоночное хранение.** Пока формат файлов JSON, в дальнейшем бинарный.
- Колоночное хранение упрощает выбор(SELECT) отдельных столбцов.

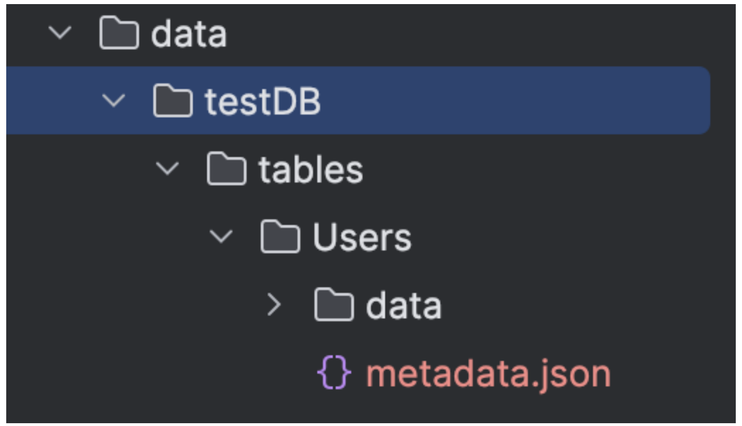
Анализ рисков

Мы уже столкнулись с некоторыми из них:

Риск	Реакция
Ошибки в механизме хранения файлов	Юнит тесты
Ошибки в парсинге запросов	Тесты и улучшение парсера
Ограниченность SQL-подмножества	Написание документации в README
Неправильная обработка ошибок	Написание своих классов исключений(Централизованный error handling)
Нехватка масщтабируемости	Модульный дизайн
Недостаточное покрытие тестами	Написание новых тестов)))

CREATE

```
CREATE TABLE Users (id INTEGER, name TEXT)
File successfully written: /Users/vladislav/NSU/SQLite-like-driver/Simple-DB-Driver/src/main/data/testDB/tables/Users/metadata.json
File successfully written: /Users/vladislav/NSU/SQLite-like-driver/Simple-DB-Driver/src/main/data/testDB/tables/Users/data/id.json
File successfully written: /Users/vladislav/NSU/SQLite-like-driver/Simple-DB-Driver/src/main/data/testDB/tables/Users/data/id.metadata.j
File successfully written: /Users/vladislav/NSU/SQLite-like-driver/Simple-DB-Driver/src/main/data/testDB/tables/Users/metadata.json
File successfully written: /Users/vladislav/NSU/SQLite-like-driver/Simple-DB-Driver/src/main/data/testDB/tables/Users/data/name.json
File successfully written: /Users/vladislav/NSU/SQLite-like-driver/Simple-DB-Driver/src/main/data/testDB/tables/Users/data/name.metadata
File successfully written: /Users/vladislav/NSU/SQLite-like-driver/Simple-DB-Driver/src/main/data/testDB/tables/Users/metadata.json
```



INSERT

```
INSERT INTO Users VALUES(1, "VLAD")
File successfully written: /Users/vladislav/NSU/SQLite-like-driver/Simple-DB-Driver/src/main/data/testDB/tables/Users/data/id.metadata.j
File successfully written: /Users/vladislav/NSU/SQLite-like-driver/Simple-DB-Driver/src/main/data/testDB/tables/Users/data/id.json
File successfully written: /Users/vladislav/NSU/SQLite-like-driver/Simple-DB-Driver/src/main/data/testDB/tables/Users/data/name.metadata
File successfully written: /Users/vladislav/NSU/SQLite-like-driver/Simple-DB-Driver/src/main/data/testDB/tables/Users/data/name.json
INSERT INTO Users VALUES(2, "Sonya")
File successfully written: /Users/vladislav/NSU/SQLite-like-driver/Simple-DB-Driver/src/main/data/testDB/tables/Users/data/id.metadata.j
File successfully written: /Users/vladislav/NSU/SQLite-like-driver/Simple-DB-Driver/src/main/data/testDB/tables/Users/data/id.json
File successfully written: /Users/vladislav/NSU/SQLite-like-driver/Simple-DB-Driver/src/main/data/testDB/tables/Users/data/name.metadata
File successfully written: /Users/vladislav/NSU/SQLite-like-driver/Simple-DB-Driver/src/main/data/testDB/tables/Users/data/name.json
```

Метадата таблицы

```json
{
  "name" : "Users",
  "columnCount" : 2,
  "countAutoIncrements" : 0,
  "countDefaults" : 0,
  "columnNames" : [ "id", "name" ]
}
```

Представление id

```json
{
  "data" : [ "1", "2" ]
}
```

Представление name

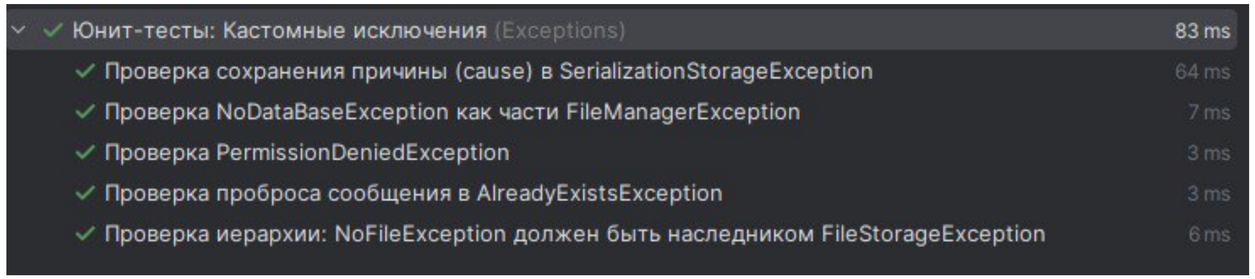
```json
{
  "data" : [ "\"VLAD\"", "\"Sonya\"" ]
}
```

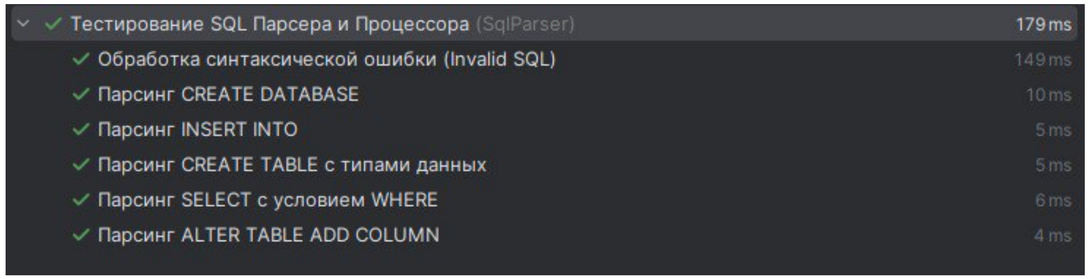
SELECT

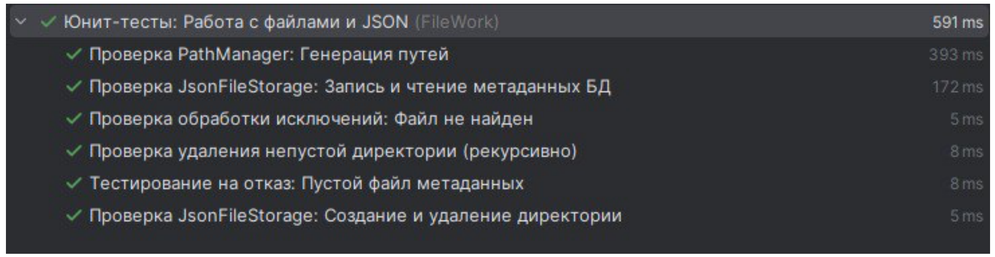
```
SELECT * FROM Users
1    "VLAD"
2    "Sonya"
SELECT id FROM Users
1
2
SELECT name FROM Users
"VLAD"
"Sonya"
```

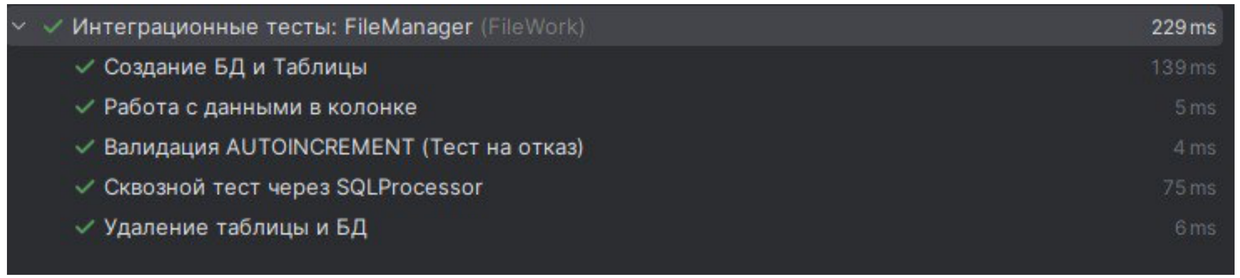
```
SELECT name, id FROM Users WHERE id=1
"VLAD"    1
SELECT name, id FROM Users WHERE name="VLAD"
"VLAD"    1
SELECT id, name FROM Users WHERE name="Sonya"
2    "Sonya"
```

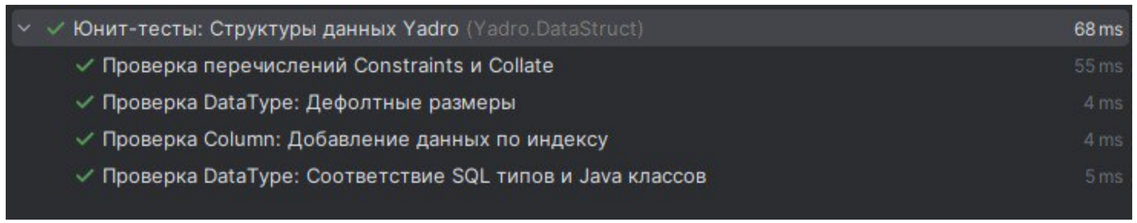
Тестирование











Демонстрация

Планы на будущее

- Переход на бинарный формат хранения данных
- Оптимизация памяти и скорости выполнения запросов

СПАСИБО ЗА ВНИМАНИЕ