Nano SQL

DeepDive Week 0 Day 3/4

Материалы и Ресурсы

За следующие несколько дней мы выполним совместное задание, в котором напишем очень простое подобие БД, а именно мы напишем простую абстракцию над нашим CSV для выполнения простых операций: *select/insert/update/delete/join*.

План такой:

1. Согласуйте, чей репозиторий будете использовать для этого задания.
2. Реализуйте вместе общую часть.
3. Поделите между собой, кто какие запросы будет делать.
4. Реализуйте запросы.
5. Обсудите как кто протестировал свои запросы.
6. Протестируйте общее решение.
7. Ревьювьте!

Общая Часть

Есть два общих класса, которые нужно создать:

```
package academy.kovalevskyi.javadeepdive.week0.day3;
import academy.kovalevskyi.javadeepdive.week0.day2.CSV;

public abstract class AbstractRequest<T> {

    protected AbstractRequest(CSV target) {
        // TODO
    }

    protected abstract T execute() throws RequestException;

    // later you can put here any protected methods that required in multiple
requests
}
```

А также одно исключение:

```
package academy.kovalevskyi.javadeepdive.week0.day3;

public class RequestException extends Exception { }
```

А также *Selector*:

```
package academy.kovalevskyi.javadeepdive.week0.day3;
public record Selector(String fieldName, String value) {

   public static class Builder {
       // TODO

       public Builder fieldName(String fieldName) {
           // TODO
       }

       public Builder value(String value) {
           // TODO
       }

       public Selector build() {
           // TODO
       }
   }
}
```

Запросы, в учебных целях, будут не мутабельные, так как данных мало и нам сейчас намного важнее сделать все потокобезопасным.

Delete Request

DeleteRequest создает новый CSV, в который копирует все записи из оригинального CSV, кроме тех, которые должны быть удалены. Строки для удаления определяются через селектор (*Selector*).

```
package academy.kovalevskyi.javadeepdive.week0.day3;
import academy.kovalevskyi.javadeepdive.week0.day2.CSV;

public class DeleteRequest extends AbstractRequest<CSV> {
    // TODO

    private DeleteRequest(CSV target, Selector whereSelector) {
        // TODO
    }

    @Override
    protected CSV execute() throws RequestException {
        // TODO
    }

    public static class Builder {
        // TODO

        public Builder where(Selector selector) {
            // TODO
        }

        public Builder from(CSV csv) {
            // TODO
        }

        public DeleteRequest build() {
            // TODO
        }
    }
}
```

Insert Request

Данный запрос создает новый CSV со всеми данными из оригинального, плюс новые записи, которые были переданы через *insert*.

```
package academy.kovalevskyi.javadeepdive.week0.day3;
import academy.kovalevskyi.javadeepdive.week0.day2.CSV;

public class InsertRequest extends AbstractRequest<CSV> {

    // TODO
    private InsertRequest(CSV target, String[] line) {
        // TODO
    }

    @Override
    protected CSV execute() throws RequestException {
        // TODO
    }

    public static class Builder {
        // TODO

        public Builder insert(String[] line) {
            // TODO
        }

        public Builder to(CSV csv) {
            // TODO
        }

        public InsertRequest build() {
            // TODO
        }
    }
}
```

Join Request

Данный запрос делает join. На CSV, с которыми делается join, накладываются следующие требования:

- оба CSV должны быть с headers
- записи в поле, по которым делается join, должны быть уникальными
- количество строк в обоих CSV должно быть равным
- на каждый уникальный ключ в одном CSV должна быть запись в другом CSV

```
package academy.kovalevskyi.javadeepdive.week0.day3;
import academy.kovalevskyi.javadeepdive.week0.day2.CSV;
public class JoinRequest extends AbstractRequest<CSV> {
    // TODO
    private JoinRequest(CSV from, CSV on, String by) {
        // TODO
    }
    @Override
    protected CSV execute() throws RequestException {
        // TODO
    }
    public static class Builder {
        // TODO
        public Builder from(CSV from) {
            // TODO
        }
        public Builder on(CSV on) {
            // TODO
        }
        public Builder by(String by) {
            // TODO
        }
        public JoinRequest build() {
            // TODO
        }
    }
}
```

Select Request

Запрос, который возвращает требуемые поля из CSV.

```
package academy.kovalevskyi.javadeepdive.week0.day3;
import academy.kovalevskyi.javadeepdive.week0.day2.CSV;

public class SelectRequest extends AbstractRequest<String[][]> {

   // TODO
   private SelectRequest(CSV target, Selector selector, String[] columns) {
       // TODO
   }

   @Override
   protected String[][] execute() throws RequestException {
       // TODO
   }
   public static class Builder {
       // TODO
       public Builder where(Selector selector) {
          // TODO
       }
       public Builder select(String[] columns) {
           // TODO
       }
       public Builder from(CSV csv) {
           this.csv = csv;
           return this;
       }
       public SelectRequest build() {
           Objects.nonNull(csv);
           return new SelectRequest(csv, selector, columns);
       }
   }
}
```

Update Request

Запрос, который создает новый CVS с новыми значениями полей, которые выбраны селектором.

```
package academy.kovalevskyi.javadeepdive.week0.day3;
import academy.kovalevskyi.javadeepdive.week0.day2.CSV;

public class UpdateRequest extends AbstractRequest<CSV> {
   // TODO
   private UpdateRequest(CSV target, Selector whereSelector, Selector
updateToSelector) {
       // TODO
   }
   @Override
   protected CSV execute() throws RequestException {
       // TODO
   }
   public static class Builder {
      // TODO
       public Builder where(Selector selector) {
           // TODO
       }
       public Builder update(Selector updateSelector) {
           // TODO
       }
       public Builder from(CSV csv) {
           // TODO
       }
       public UpdateRequest build() {
           // TODO
       }
   }
}
```

SQL Parser

Эта опциональная часть которая пока что НЕ покрыта тестами и не входит в ScoringCard. Однако, если все уже закончено, а до следующего дня еще есть время, реализуйте программу “SQL parser”, которая будет считывать из *stdin* SQL команду и выполнять её. SQL команда будет упрощенной и имеет следующий вид:

```
SELECT|UPDATE|DELETE
FROM <csv_file_name>
[VALUES ...]
[WHERE ...]
```