Санкт-Петербургский Политехнический Университет Петра Великого

Институт компьютерных наук и технологий

Кафедра компьютерных систем и программных технологий

ОТЧЕТ
по лабораторной работе

«Генератор тестовых данных»

Базы данных

Работу выполнил студент

группа 43501/3 Дьячков В.В.

Работу принял преподаватель

_____ Мяснов А.В.

Санкт-Петербург

2018

Содержание

1. Цель работы

Сформировать набор данных, позволяющий производить операции на реальных объемах данных.

2. Программа работы

1. Реализация в виде программы параметризуемого генератора, который позволит сформировать набор связанных данных в каждой таблице.
2. Частные требования к генератору, набору данных и результирующему набору данных:
   - количество записей в справочных таблицах должно соответствовать ограничениям предметной области;
   - количество записей в таблицах, хранящих информацию об объектах или субъектах должно быть параметром генерации;
   - значения для внешних ключей брать из связанных таблиц.

3. Выполнение работы

3.1. Выбор технологий для реализация генератора

Для реализации генератора был использован язык программирования **Java**.

Для взаимодействия Java с базой данных используется технология **JDBC** (Java Database Connectivity), которая обеспечивает доступ Java API к реляционным базам данных (обеспечивает соединение с БД) и позволяет создавать SQL-выражения, выполнять SQL-запросы, просматривать и модифицировать записи.

Для доступа к каждой конкретной БД необходим специальный JDBC-драйвер, который является адаптером Java-приложения к БД. Например, для доступа к базе данных PostgreSQL используется драйвер, который предоставляет PostgreSQL.

Минусом использования JDBC напрямую является необходимость написания большого количества однообразного кода и ручного формирования SQL-запросов для выборки и изменения данных. Вместо этого была выбрана технология **ORM** (Object-Relational Mapping), которая связывает базы данных с концепциями объектно-ориентированных языков, создавая «виртуальную объектную базу данных». Самым распространенным ORM-фреймворком для Java является **Hibernate**.

ORM является еще одним уровнем абстракции, построенным поверх JDBC. Hibernate позволяет отобразить сущности, хранящиеся в РБД, на Java-

классы, упростив при этом выборку и обновление данных. С помощью конфигурационных файлов и Java-аннотаций можно указать Hibernate как извлекать данные из класса и соединять с определенным столбцами в таблице БД. Кроме этого в конфигурационном файле указывается диалект базы данных (в нашем случае `org.hibernate.dialect.PostgreSQL9Dialect`), класс JDBC драйвера (`org.postgresql.Driver`), URL, имя пользователя и пароль к базе данных и др.

В Hibernate для получения физического соединения с базой данных используется сессия (session). Благодаря тому, что сессия является легковесны объектом. С помощью сессий появляется возможность создавать, читать, редактировать и удалять объекты внутри базы данных.

3.2. Реализация сущностей прикладной области

Для работы с Hibernate необходимо создать Persistent Classes, чтобы отобразить на них сущности из реляционной базы данных. Такие классы удобно реализовать в виде POJO (Plain Old Java Object) классов, содержащих пустой конструктор, private поля, соответствующие полям соответствующий таблицы РБД, и get/set методы для них. Помимо этого внутри данных классов используются аннотации из пакета `javax.persistence`:

- `@Entity` для пометки POJO класса как JPA (Java Persistence API) сущности;
- `@Table` для указание имени соответствующей таблицы базы данных;
- `@Id` для пометки первичного ключа таблицы;
- `@GeneratedValue` используется совестно с `@Id` и указывает на то, что ключ генерируется базой данных;
- `@Column` для указания имени, длины, nullability и уникальности соответствующего столбца таблицы;
- `@OneToOne`, `@OneToMany`, `@ManyToOne` и `@ManyToMany` для указания соответствующих связей между сущностями и др.

Все POJO классы прикладной области (бронирование отелей), такие как `User`, `Hotel`, `Reservation` и др. были реализованы на языке Java внутри пакета `com.vaddya.hotelbooking.model`.

3.3. Реализация сущностей доступа к данным

Для описания доступа к данным был разработан параметризованный интерфейс `Dao<E, I extends Serializable>`, в котором указаны методы, которые необходимо реализовать для манипулирования некоторой сущностью прикладной области `E`, обладающей ключом с типом `I`.

```java
package com.vaddya.hotelbooking.dao;

import java.io.Serializable;
import java.util.List;

public interface Dao<E, I extends Serializable> {

    boolean insert(E t);

    E find(I id);

    List<E> findAll();

    boolean update(E t);

    void delete(I id);
}
```

Листинг 1: `Dao.java`

Общая для всех сущностей логика была реализована в абстрактном классе `EntityDao`, внутри которого используется объект сессии для получения и обновления данных с использованием транзакций. Для всех сущностей прикладной области внутри пакета `com.vaddya.hotelbooking.dao` были созданы DAO (Data Access Object) классы, являющиеся наследниками `EntityDao`. С их помощью прикладная программа (генератор), взаимодействует с базой данных.

3.4. Реализация генератора тестовых данных

Для генерации тестовых данных была использована библиотека **Faker**, которая является портированной на Java библиотекой для языка Ruby. Faker позволяет генерировать осмысленные значения для полей имени, адреса, телефона и др. Логика создания случайных тестовых объектов для каждой сущности прикладной области была реализована в классе `EntityGenerator` в пакете `com.vaddya.hotelbooking.generator`. В том же пакете был реализован класс `HibernateSessionFactory` для настройки фабрики Hibernate сессий.

В классах `GeneratorOptions` и `Generator` реализована основная логика работы генератора. Внутри `GeneratorOptions` используется библиотека Apache Commons CLI для удобного разбора аргументов командой строки генератора и формирования поясняющего вывода. Генерирование данных может происходить в нескольких режимах в зависимости от параметров командой строки:

- `-a,--all` – генерация всех сущностей;
- `-c,--cluster` – «кластерная» генерация данных:
    1. `city` – генерация только стран, городов и пользователей;

2. `hotel` – генерация только отелей и их правил, тип комнат и удобств в них, комнат и цен;
3. `reservation` – генерация только бронирований и бонусов/штрафов, гостей, отмен и отзывов к ним.

Для каждого из способов можно указать только `-n,--number` – базовое число, относительно котрого будет автоматически выбрано соответствующее число элементов той или иной сущности. При этом если не указать его или число генерируемых данных, то будет использовано значение по умолчанию.

```
usage: generator [-?] [-a] [-b <arg>] [-c <arg>] [-f <arg>] [-g] [-h <arg>]
[-i <arg>] [-l <arg>] [--max-bp <arg>] [--max-facilities <arg>]
[--min-bp <arg>] [--min-facilities <arg>] [-n <arg>] [-o <arg>]
[-p] [-r <arg>] [-s <arg>] [-t] [-u <arg>] [-v <arg>] [-w <arg>]
    -?,--help                    print help
    -a,--all                     generate all entities
    -b,--bonus-penalties <arg>   number of bonuses or penalties, default: 20
    -c,--cluster <arg>           cluster type: [city | hotel | reservation]
    -f,--facilities <arg>        number of facilities, default: 100
    -g,--guests                  generate guests
    -h,--hotels <arg>            number of hotels, default: 50
    -i,--cities <arg>            number of cities, default: 100
    -l,--cancellations <arg>     number of cancellations, default: 100
    --max-bp <arg>               maximum number of bonuses or penalties per
        reservation, default: 2
    --max-facilities <arg>       maximum number of facilities per room,
        default 30
    --min-bp <arg>               minimum number of bonuses or penalties per
        reservation, default: 0
    --min-facilities <arg>       minimum number of facilities per room,
        default 10
    -n,--number <arg>            base number of cluster, default 1000
    -o,--counties <arg>          number of countries, default: 10
    -p,--prices                  generate prices for room types
    -r,--rooms <arg>             number of rooms, default: 1000
    -s,--house-rules <arg>       number of house rules, default: 250
    -t,--room-types              generate room types
    -u,--users <arg>             number of users, default: 5000
    -v,--reservations <arg>      number of reservations, default: 50000
    -w,--reviews <arg>           number of reviews, default: 20000
```

В классе `Generator` производится анализ аргументов командой строки, после чего с помощью `EntityGenerator` происходит генерирование нужного количества сущностей каждого типа, которые вставляются в базу данных при помощи DAO классов.

4. Выводы

В процессе данной работы был разработан генератор тестовых данных на языке Java с использование ORM-библиотеки Hibernate. ORM подход позволяет избежать ручного написания SQL-запросов при помощи отображения таблиц реляционной базы данных на Java-классы.