Задание Hibernate Core

Автор - Павел Сорокин

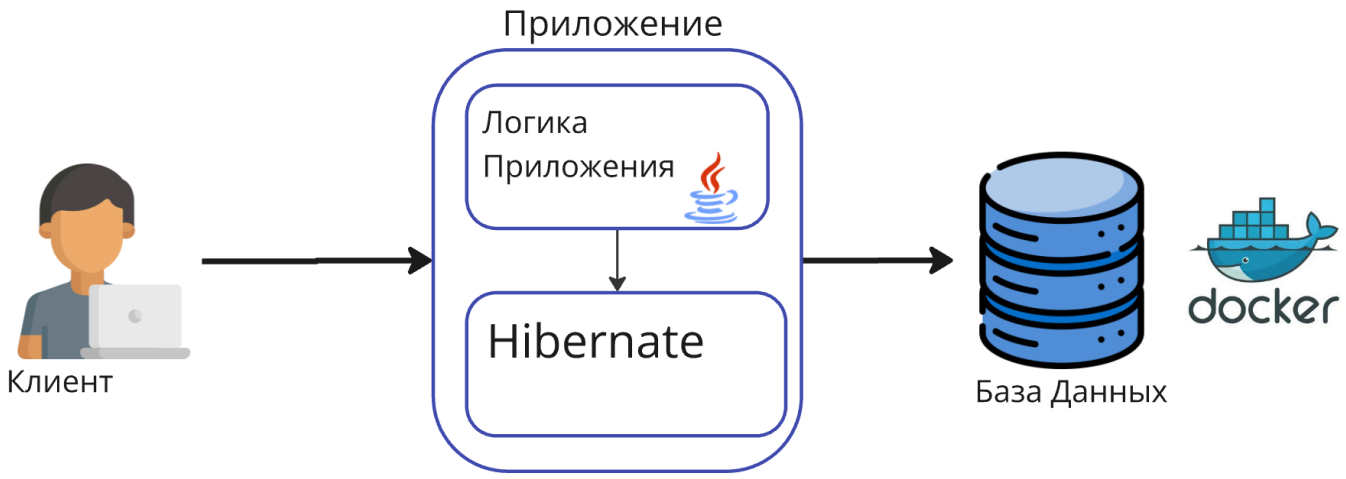
Описание проекта

Что будет сделано:
Вы разработаете консольное приложение для управления клиентами, их заказами и купонами. Приложение будет позволять:

- **Работа с клиентами:**
  - Добавление клиента вместе с его профилем, начальными заказами и привязкой купонов.
  - Удаление клиента (при этом все связанные сущности, такие как профиль, заказы, будут корректно удаляться с применением каскада).
- **Методы для работы с отдельными сущностями:**
  - **Добавление заказа:** Отдельный метод для добавления нового заказа существующему клиенту.
  - **Изменение профиля:** Отдельный метод для изменения данных профиля клиента.
- **Редактирование купонов:**

    - Метод для изменения информации по купонам (например, обновление кода или процента скидки).
- **Поиск заказов:**
    - Функция поиска заказов по заданным фильтрам с использованием кастомных запросов (JPQL и/или нативного SQL).
- **Решение проблемы N+1:**
    - В качестве продвинутой темы студент может реализовать выборку данных с оптимизацией (например, через fetch join), чтобы устранить проблему N+1 запросов.

Ваше приложение:



Необходимые навыки для реализации

Вы должны разбираться в следующих темах:

- **Основы Hibernate:** Конфигурация, настройка сущностей, жизненный цикл объектов, работа с Session.
- **Маппинг связей:** Создание и настройка связей один-к-одному, один-ко-многим, многие-ко-многим. Использование аннотаций `@OneToOne`, `@OneToMany`, `@ManyToOne` и `@ManyToMany`, а также `@JoinTable` для реализации вспомогательной таблицы.
- **Работа с PostgreSQL:** Настройка подключения, генерация схемы, использование каскадных операций при сохранении/удалении.
- **Кастомные запросы:** Реализация JPQL и/или SQL запросов для фильтрации и сложных выборок.

- **Создание консольного приложения:** Построение меню, ввод данных, делегирование работы Hibernate.
- **Оптимизация запросов:** Знание проблемы N+1 и методов её устранения (например, через fetch join).

Детальное описание задания

Структура проекта и основные сущности

Проект будет состоять из пяти таблиц, отражающих следующие сущности и связи:

Client

- Поля: `id` (PK), `name`, `email`, `registrationDate`
- Связи:
  - One-to-one с **Profile**
  - One-to-many с **Order**
  - Many-to-many с **Coupon** (через таблицу **CLIENT_COUPONS**)

Profile

- Поля: `id` (PK), `address`, `phone`
- Связи:
  - One-to-one с **Client**

Order

- Поля: `id` (PK), `orderDate`, `totalAmount`, `status`
- Связи:
  - Many-to-one с **Client**

Coupon

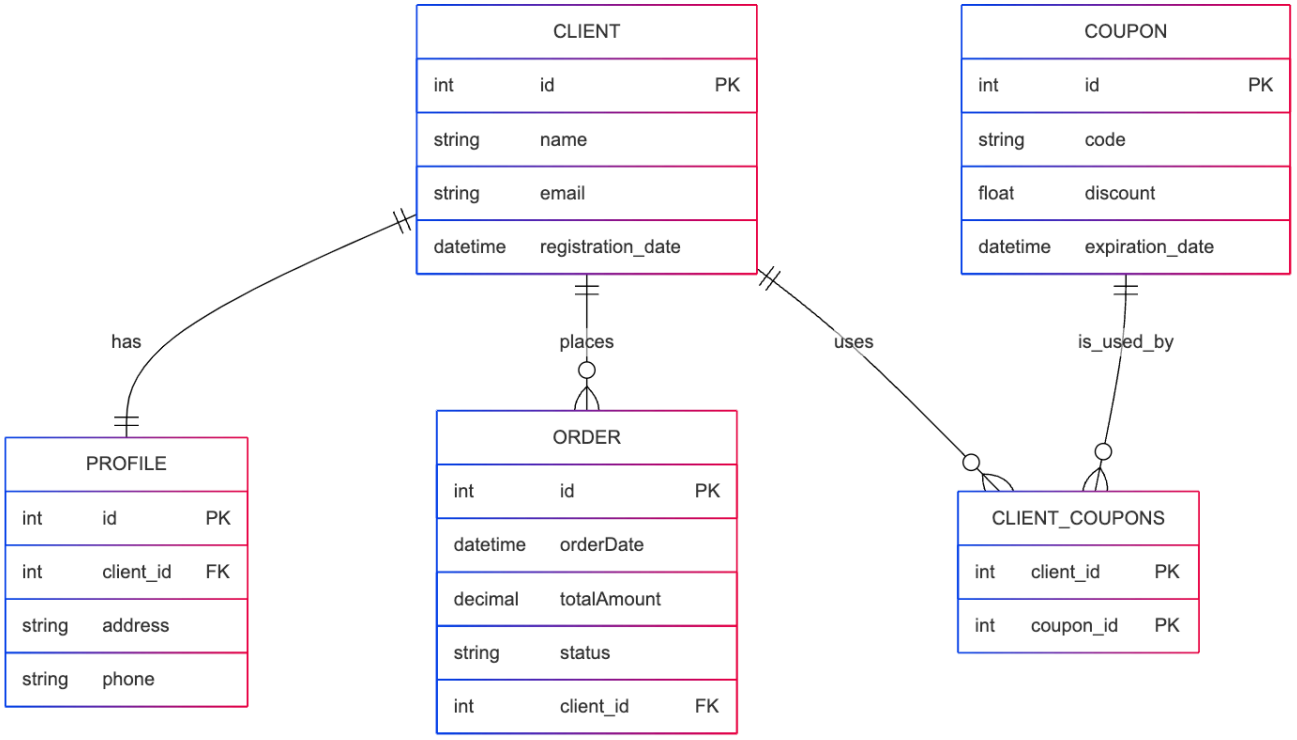
- Поля: `id` (PK), `code`, `discount`
- Связи:
  - Many-to-many с **Client**

CLIENT_COUPONS (вспомогательная таблица)

- Поля: `client_id`, `coupon_id`

- Служит для организации связи многие-ко-многим между **Client** и **Coupon**.

Итоговая структура таблиц:



Функциональные возможности и методы приложения

Приложение должно включать следующие возможности:

1. **Добавление клиента:**
   - Ввод имени, email.
   - Автоматическое создание связанного профиля (ввод адреса, телефона).
   - Возможность добавить начальный заказ (при желании) и привязать купоны.
   - При сохранении клиент и все связанные объекты сохраняются через Hibernate с применением каскадных операций.
2. **Удаление клиента:**
   - Удаление клиента вместе с каскадным удалением профиля и заказов.
   - Проверка корректности работы связей, чтобы данные из вспомогательной таблицы **CLIENT_COUPONS** также корректно удалялись.
3. **Редактирование купонов:**

- Выбор купона по ID.
- Внесение изменений в поля купона (например, код, значение скидки).
- Сохранение изменений через Hibernate, без нарушения связей many-to-many.

4. **Поиск заказов:**
   - Ввод фильтров: дата, сумма, статус.
   - Реализация кастомного JPQL/SQL запроса для поиска заказов.
   - Вывод результатов в консоль в удобном для чтения формате.
5. **Добавление заказа:**
   - Позволяет добавить новый заказ для уже существующего клиента.
   - Ввод данных заказа: дата, сумма, статус и другие атрибуты.
   - Сохранение нового заказа с указанием связи к клиенту.
6. **Изменение профиля:**
   - Позволяет изменить данные профиля существующего клиента.
   - Ввод новых значений для полей профиля (например, адрес, телефон).
   - Сохранение обновлений через Hibernate с корректной привязкой к клиенту.
7. **Решение проблемы N+1 (опционально, продвинутая тема):**
   - Реализация выборки клиентов вместе с заказами с использованием fetch join для устранения проблемы N+1 запросов.
   - Тестирование и анализ логов SQL в консоли.

Рекомендации по реализации

Основной порядок выполнения

1. **Подготовка проекта:**
   - Инициализируйте Maven/Gradle-проект.
   - Подключите зависимости для Hibernate, PostgreSQL и других нужных библиотек.
   - Создайте файл конфигурации Hibernate (например, `hibernate.cfg.xml`) или используйте Java-конфигурацию.

2. **Реализация сущностей:**
   - Создайте классы для всех сущностей: Client, Profile, Order, Coupon.
   - Настройте поля с аннотациями (`@Entity` , `@Table` , `@Column` , `@Id`).
   - Определите связи между сущностями, включая вспомогательную таблицу **CLIENT_COUPONS** для связи Many-to-Many.
3. **Настройка операций (CRUD):**
   - Реализуйте методы добавления/удаления клиента, учитывая каскадные операции для профиля и заказов.
   - Добавьте метод для редактирования профиля. Используйте методы сессии для поиска клиента по id и обновления его профиля.
   - Добавьте метод для добавления нового заказа к существующему клиенту.
   - Реализуйте метод для редактирования купонов.
4. **Реализация кастомных запросов:**
   - Создайте кастомные запросы на JPQL или нативном SQL для:
     - Поиска заказов по фильтрам (например, по дате, сумме, статусу).
     - Выборки клиентов и их заказов с применением fetch join для устранения N+1.
   - Добавьте логирование SQL-запросов для отладки и проверки корректности выборок.
5. **Разработка консольного интерфейса:**
   - Организуйте главное меню с выбором команд:

     Меню приложения

     ```
     Выберите действие:
     1. Добавить клиента
     2. Удалить клиента
     3. Редактировать профиль
     4. Добавить заказ
     5. Редактировать купоны
     6. Найти заказы
     7. Выход

     Введите номер команды:
     ```

   - Реализуйте ввод данных с консоли и соответствующий вызов методов для работы с Hibernate.
6. **Тестирование и отладка:**

- Проверьте корректность операций добавления, удаления, редактирования и поиска.
- Используйте SQL-логи Hibernate для отслеживания проблем N+1.
- Прокомментируйте логику кастомных запросов для будущего анализа.

Пример работы приложения

При запуске:
Отобразится главное меню:

главное меню

```
Выберите действие:
1. Добавить клиента
2. Удалить клиента
3. Редактировать профиль
4. Добавить заказ
5. Редактировать купоны
6. Найти заказы
7. Выход

Введите номер команды:
```

Добавление клиента:
Введите:

- Имя, email;
- Данные профиля (адрес, телефон);
- Выбор из доступных купонов.

После сохранения вы получите подтверждение с ID созданного клиента.

Редактирование профиля:

- Выберите клиента по ID
- Введите новые данные для профиля (например, новый адрес или телефон)
- После успешного сохранения выведите сообщение об успешном обновлении.

Добавление заказа:

- Выберите существующего клиента
- Введите данные нового заказа (orderDate, totalAmount, status)

- После сохранения приложение выведет подтверждение создания заказа с его ID.

Редактирование купонов и поиск заказов:

- Используйте соответствующие пункты меню для редактирования купонов и поиска заказов с фильтрами по сумме, дате или статусу.

Тестирование и советы

- **Проверка каскадного удаления:**
  - Добавьте клиента с несколькими заказами и профилем.
  - Удалите клиента и проверьте, что профиль и заказы удалены (просмотрите содержимое таблиц в БД).
- **Проверка редактирования профиля и купонов:**
  - Отредактируйте профиль клиента и купоны, затем выполните выборку данных из БД, чтобы убедиться, что изменения сохранены.
- **Проверка добавления заказа:**
  - Добавьте несколько заказов для одного клиента, после чего выполните выборку заказов с использованием кастомного запроса.
- **Проверка кастомных запросов:**
  - Используйте фильтры для поиска заказов (например, по дате или сумме) и анализируйте вывод SQL-логов для проверки оптимизации (fetch join для решения проблемы N+1).
- **Логирование SQL:**
  - Включите логирование SQL запросов Hibernate для отслеживания работы кастомных запросов и выявления возможных проблем с N+1.