УДК 004

Написание подзапросов EXISTS с помощью JPA и Hibernate

Еровлева Регина Викторовна
Приамурский государственный университет имени Шолом-Алейхема
Студент

Еровлев Павел Андреевич
Приамурский государственный университет имени Шолом-Алейхема
Студент

Аннотация
В данной статье описывается процесс использования подзапросов с помощью Spring Data JPA и Hibernate. Разобраны примеры написания запросов. Работа происходит в среде программирования intelijIdea.
Ключевые слова: Spring Boot, Spring Data, JPA

Writing EXISTS subqueries with JPA and Hibernate

Erovleva Regina Viktorovna
Sholom-Aleichem Priamursky State University
Student

Erovlev Pavel Andreevich
Sholom-Aleichem Priamursky State University
Student

Abstract
This article describes the process of using subqueries with Spring Data JPA and Hibernate. Analyzed examples of writing queries. The work takes place in the intelijIdea programming environment.
Keywords: Spring Boot, Spring Data, JPA

Spring Data JPA - предоставляет знакомую и последовательную модель программирования на основе Spring для доступа к данным, сохраняя при этом особые черты базового хранилища данных.

Подзапросы EXISTS очень полезны, так как позволяют реализовать SemiJoins.

В.Л. Волушкова в своей работе привела примеры разработки технологий на примере языка Java [1]. А.А Аббакумов., А.И.Егунова, Ю.С.Вечканова, М.А.Воропаева, рассказывают о проблемах повышения эффективности образовательной деятельности вуза. Они предложили реализацию информационного хранилища в виде J2EE-приложения с использованием фреймворка Spring [2]. В своей работе В.С.Жданова

рассмотрела подход в создании серверной части мобильного приложения для расписания [3]. П.В.Прохоров, Н.В.Разговоров рассмотрели технологии и современные подходы в разработке приложений на примере онлайн-магазина с использованием Spring [4]. Так же Н.Е.Губенко, А.Д.Нарижный провели сравнительный анализ технологий, которые имеют схожую функциональность и которые предназначены для одних и тех же задач. [5].

Цель исследования – провести работу с подзапросами Exists с помощью Spring Data JPA и Hibernate.

Будем использовать следующие объекты «Post» и «PostComment» объекты(рис.1).

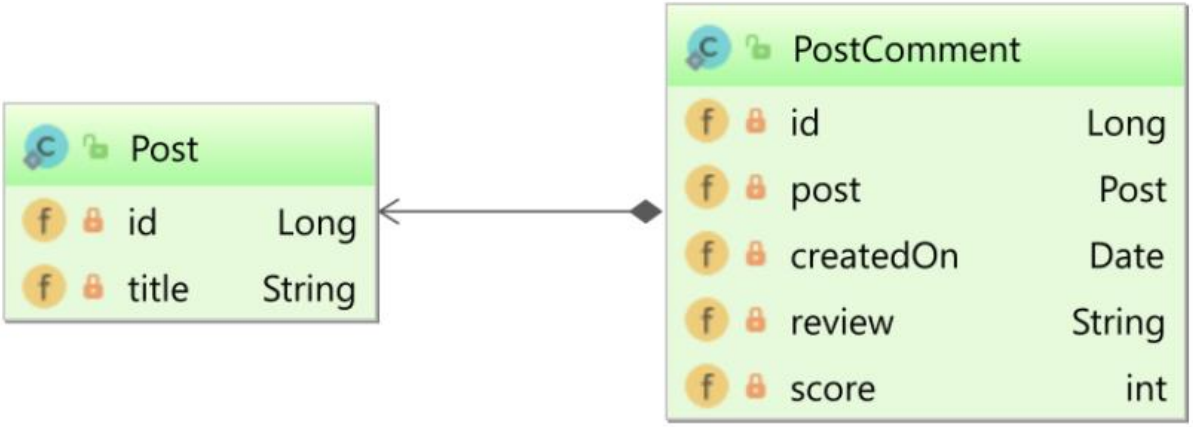

Рисунок 1 – Запись БД

Объект «Post» является родительским, а объект «PostComment» является дочерним, поскольку «PostComment» ссылается на родительский объект через его «post» свойство.

Теперь попробуем получить все «Post» сущности в «PostComment», где счетчик больше 10 (рис.2).

```
List<Post> posts = entityManager.createQuery("""
    select distinct p
    from PostComment pc
    join pc.post p
    where pc.score > :minScore
    order by p.id
    """, Post.class)
.setParameter("minScore", 10)
.setHint(QueryHints.HINT_PASS_DISTINCT_THROUGH, false)
.getResultList();
```

Рисунок 2 – Запрос на вывод сущностей

Этот запрос выполняет соединение между «post» и «post_comment» только для фильтрации «post» записей. Поскольку проекция содержит только «Post» объект, в этом случае JOIN не требуется. Вместо этого для фильтрации записей сущностей следует использовать «SemiJoin Post».

Подзапрос «EXISTS» гораздо лучшая альтернатива. Для этого обновим предыдущий запрос (рис.3).

```
List<Post> posts = entityManager.createQuery("""
    select p
    from Post p
    where exists (
       select 1
       from PostComment pc
       where
          pc.post = p and
          pc.score > :minScore
    )
    order by p.id
    """, Post.class)
.setParameter("minScore", 10)
.getResultList();
```

Рисунок 3 – Запрос EXISTS

При выполнении приведенного выше запроса «JPQL» «Hibernate» генерирует запрос SQL (рис.4).

```
SELECT
    p.id AS id1_0_,
    p.title AS title2_0_
FROM post p
WHERE EXISTS (
    SELECT 1
    FROM post_comment pc
    WHERE
        pc.post_id=p.id AND
        pc.score > ?
)
ORDER BY p.id
```

Рисунок 4 – SQL запрос

Преимущество данного запроса в том, что «SemiJoin» не нужно объединять все записи «post» и «post_comment», поскольку, как только «post_comment» найдено в соответствие критериям фильтрации, «EXISTS» возвращает «true» и запрос переходит к следующей «post» записи.

Предыдущий запрос «JPQL» можно переписать в запрос «Criteria API» (рис.5).

```
CriteriaBuilder builder = entityManager.getCriteriaBuilder();

CriteriaQuery<Post> query = builder.createQuery(Post.class);
Root<Post> p = query.from(Post.class);

ParameterExpression<Integer> minScore = builder.parameter(Integer.class);
Subquery<Integer> subQuery = query.subquery(Integer.class);
Root<PostComment> pc = subQuery.from(PostComment.class);
subQuery
    .select(builder.literal(1))
    .where(
        builder.equal(pc.get(PostComment_.POST), p),
        builder.gt(pc.get(PostComment_.SCORE), minScore)
    );

query.where(builder.exists(subQuery));

List<Post> posts = entityManager.createQuery(query)
    .setParameter(minScore, 10)
    .getResultList();
```

Рисунок 5 – Запрос с использованием Criteria API

Так же существует «Blaze Persistence», который позволяет написать динамические запросы, которые не только более удобочитаемы, но и более эффективны, поскольку позволяет использовать «LATERAL JOIN», «Derived Tables», «Common Table Expressions» или «Window Functions».

Предыдущий запрос «Criteria API» можно переписать с использованием «Blaze Persistence» (рис.6).

```
final String POST_ALIAS = "p";
final String POST_COMMENT_ALIAS = "pc";

List<Post> posts = cbf.create(entityManager, Post.class)
    .from(Post.class, POST_ALIAS)
    .whereExists()
        .from(PostComment.class, POST_COMMENT_ALIAS)
        .select("1")
        .where(PostComment_.POST).eqExpression(POST_ALIAS)
        .where(PostComment_.SCORE).gtExpression(":minScore")
    .end()
    .select(POST_ALIAS)
    .setParameter("minScore", 10)
    .getResultList();
```

Рисунок 6 – Запрос с использованием Blaze Persistence

В SQL «SemiJoins» выражаются с помощью подзапросов «EXISTS» и эта функция не ограничивается собственными SQL-запросами, поскольку можно использовать «EXISTS» в запросах сущностей «JPA» и «Hibernate» как с «JPQL», так и с «Criteria API», а также с запросами «Blaze Persistence».

Библиографический список

1. Волушкова В.Л. Архитектурные решения java для доступа к данным // Теоретические основы программирования. Учебное пособие. Тверской государственный университет, 2019. 137с.
2. Егунова А.И., Аббакумов А.А., Воропаева М.А., Вечканова Ю.С. Система управления хранилищем электронных образовательных ресурсов // Образовательные технологии и общество. 2019. №3. С. 145-154.
3. Жданова В.С. Серверный модуль системы уведомления об изменениях в расписании занятий // Молодость. Интеллект. Инициатива. 2017. С. 21-22.
4. Прохоров П.В., Разговоров Н.В. Современные подходы в backend разработке на примере онлайн-магазина // Прикладная математика и фундаментальная информатика. 2020. №2. С. 23-28.
1. Нарижный А.Д., Губенко Н.Е. Сравнительный анализ стеков технологий spring и javaee (jakartaee) для разработки enterprise приложений // Информатика, управляющие системы, математическое и компьютерное моделирование 2020. №3. С. 549-462.