И. С. Клезович, Е. В. Рафалова
(ГГУ имени Ф. Скорины, Гомель)

ИСПОЛЬЗОВАНИЕ МЕТОДОВ МУТАЦИИ SPRING DATA ДЛЯ ПОСТРОЕНИЯ SQL-ЗАПРОСОВ

Spring Data JPA реализует декларативный подход к формированию SQL-запросов посредством анализа сигнатур методов в интерфейсах репозиториев. Данный механизм основан на конвенциональном парсинге имён методов, что позволяет абстрагироваться от явного написания JPQL или SQL-кода. Процесс трансляции включает фазы, представленные на рисунке 1.

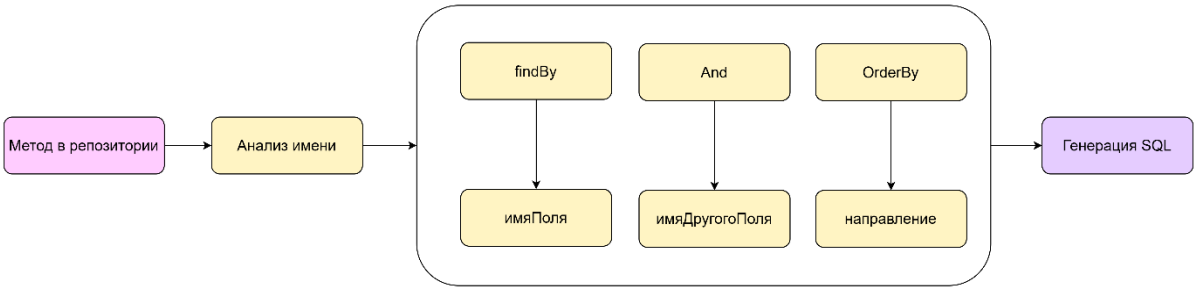

Рисунок 1 – Механизм преобразования имени метода в SQL

Простым примером механизма является формирование запроса на выборку по параметру. Класс UserRepository наследует функциональность JpaRepository<User, Long>, где User – сущность JPA, связанная с таблицей user в БД, а Long — тип первичного ключа сущности. Метод findByEmail декларирует запрос для выборки данных. Токен findBy – индикатор операции выборки (аналог SELECT в SQL). Токен Email — ссылка на атрибут email сущности User. Имя атрибута должно строго соответствовать полю сущности (с учётом регистра CamelCase). Код метода представлен в листинге 1.

Листинг 1

```
public interface UserRepository extends JpaRepository<User, Long> {
    List<User> findByEmail(String email);
}
//SELECT * FROM user WHERE email = ?
```

Принцип генерации запроса включает 3 шага: семантический разбор имени метода, формирование предиката, трансляция в SQL.

Семантический разбор имени метода Spring Data JPA выполняет лексическую декомпозицию на ключевые слова (findBy) и атрибуты сущности (Email). Далее происходит валидация существования атрибута email в классе User.

Формирование предиката происходит посредством генерации условия фильтрации: WHERE email = ?. Параметр ? связывается с аргументом метода String email через позиционное соответствие.

Последний этап генерации запроса заключается в трансляции полученных данных в SQL. Для сущности User, размеченной аннотацией @Entity, ORM (Hibernate) преобразует запрос в: SELECT * FROM user WHERE email = ?. Нужно иметь в виду, что изначально генерируется JPQL-запрос SELECT u FROM User u WHERE u.email = :email, который адаптируется к диалекту используемой СУБД.