Use native queries to perform bulk updates

Bulk updates provide Performance benefits

With the standard JPA approach, you fetch an entity from the database and call some setter methods to update it.

This requires at least 2 SQL statements for each entity and can create performance issues if you work on a huge set of entities. It's often a lot faster to update all entities with one native UPDATE statement.

```
em.createNativeQuery(
        "UPDATE person p SET firstname = firstname || '-changed'")
        .executeUpdate();
```

Problem 1: Outdated 1st level cache

Hibernate doesn't know which records the native query updates and can't update or remove the corresponding entities from the first level cache.

You either need to make sure that you didn't fetch any entities that will be affected by the UPDATE statement or remove these entities from the cache yourself.

You can do this by calling the *detach()* method. But before you do that, make sure to call the *flush()* method to write all changes to the database.

```
PersonEntity p = em.find(PersonEntity.class, 1L);


log.info("Detach PersonEntity");
em.flush();
em.detach(p);


em.createNativeQuery(
        "UPDATE person p SET firstname = firstname || '-changed'")
        .executeUpdate();
```

Problem 2: Not part of the entity life cycle

The native UPDATE statement is executed in the database and doesn't use any entities. This provides performance benefits, but it also avoids the execution of any entity lifecycle methods or entity listeners.

If you use a framework like Hibernate Envers or implement any code yourself that relies on lifecycle events, you have to either avoid native UPDATE statements or implement the operations of your listeners within this specific use case.