Hibernate Tip: What is the fastest option to delete 100 database records

Question:

In one of my use cases, I need to remove a few hundred entities, and I can see in my tests that this gets really slow. Is there any better option than the *EntityManager.remove* method?

Solution:

JPA and Hibernate provide 2 general options to remove entities.

The most common one is the *remove* method of the *EntityManager*. It is easy to use but also very inefficient if you need to remove a list of entities.

For each of the entities you want to delete, Hibernate has to load the entity, perform the lifecycle transition to *removed* and trigger the SQL DELETE operation. That requires at least 1 query to load all entities and an additional SQL DELETE statement for each of them. So, when you use this approach to remove 100 entities, Hibernate has to perform at least 101 SQL statements.

It's often better to remove such a list of entities with a JPQL query. Similar to SQL, you can define a JPQL DELETE operation that removes all entities with just 1 statement.

```
// Write all pending changes to the DB and clear persistence
context
em.flush();
em.clear();

// Remove all entities referenced in the List ids variable
Query query = em.createQuery("DELETE Author a WHERE
id IN (:ids)");
query.setParameter("ids", ids);
query.executeUpdate();
```

But be careful, this approach has a drawback. Hibernate doesn't load the entities and doesn't perform the lifecycle state transition to *removed*. That provides substantial performance benefits, but it also prevents Hibernate from updating its caches, incl. the 1st level cache, and from triggering any lifecycle events or listeners.

You should always call the *flush* and *clear* method on your *EntityManager* to detach all entities from the current persistence context and to clear the 1st level cache before you perform a JPQL UPDATE or DELETE statement. I get into more details on that in Hibernate Tips: How to remove entities from the persistence context before doing bulk operations.

Learn more

You can use the same approach with Native Queries or the Criteria API.

Hibernate Tips Book