# SQL Result Set Mappings

One downside of native queries is that they return a *List* of *Object[]* instead of the mapped entities and value objects you normally use. Each of the *Object[]* represents one record returned by the database.

You then need to iterate through the array, cast each Object to its particular type and map them to our domain model. This creates lots of repetitive code and type casts as you can see in the following example.

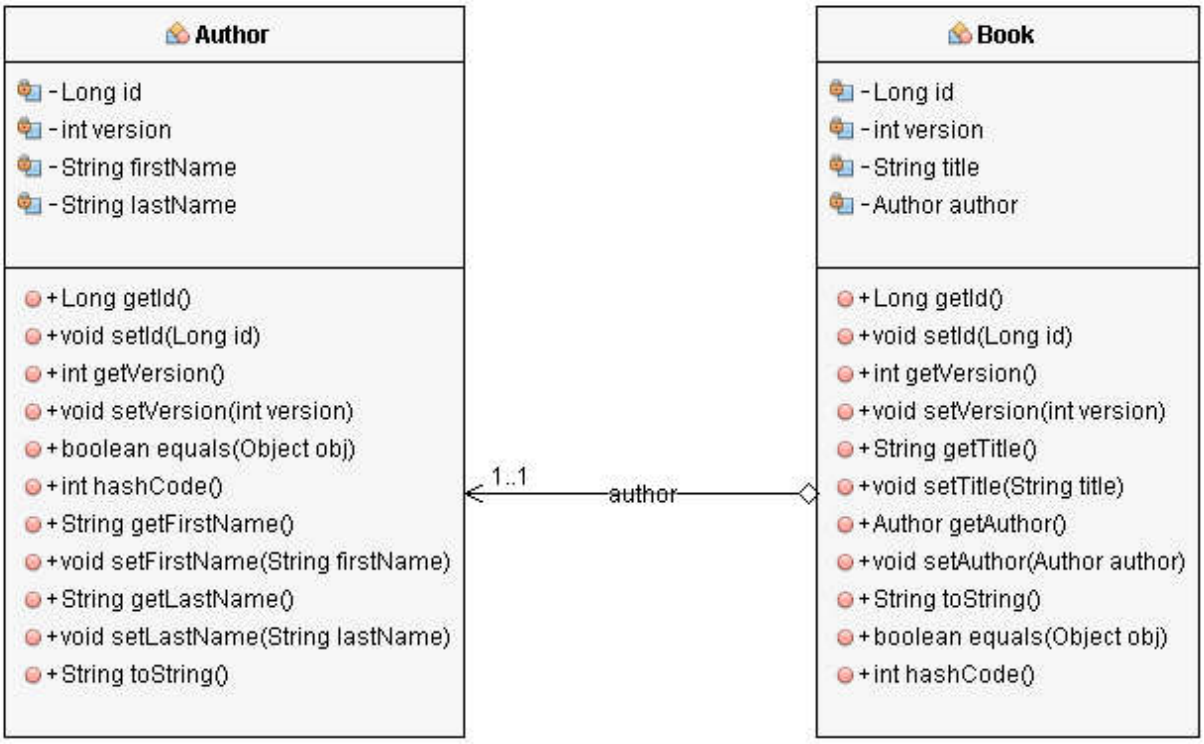
```
List<Object[]> results = this.em.createNativeQuery("SELECT a.id, a.firstName, a.\
lastName, a.version FROM Author a").getResultList();

results.stream().forEach((record) -> {
        Long id = ((BigInteger) record[0]).longValue();
        String firstName = (String) record[1];
        String lastName = (String) record[2];
        Integer version = (Integer) record[3];
});
```

It would be more comfortable if we could tell the *EntityManager* to map the result of the query into entities or value objects as it is the case for JPQL statements. The good news is, JPA provides this functionality. It is called SQL result set mapping, and I will explain it in this chapter.

## The example

Before we dive into the details of SQL result set mappings, let's have a look at the entity model that we will use during this chapter.

It consists of an *Author* entity with an id, a version, a first name and a last name. For the more complex mapping examples, I also need the *Book* entity which has an id, a version, a title and a reference to the *Author*. To keep it simple, each book is written by only one author.




**Class diagram entites**

# Basic mappings

Let's begin with some basic mappings that tell Hibernate to map each record of the result set to an entity. The easiest way to do that is to use the default mapping, but you can also provide your own mapping definition.

## How to use the default mapping

You just need to provide the entity class as a parameter to the *createNativeQuery(String sqlString, Class resultClass)* method of the *EntityManager* to use the default mapping. The following snippet shows how this is done with a very simple query. In a real project, you would use this with a stored procedure or a very complex SQL query.

```
List<Author> results = this.em.createNativeQuery("SELECT a.id, a.firstName, a.la\
stName, a.version FROM Author a", Author.class).getResultList();
```

The query needs to return all attributes of the entity, and the JPA implementation (e.g. Hibernate) will try to map the returned columns to the entity attributes based on their name and type. The