**N+1 Problem in Hibernate and its solution-Precise-2022**

The N+1 loading problem occurs when an application makes one query to retrieve the initial data (e.g., a list of users) and then an additional query for each row of data to retrieve related data (e.g., the user’s articles) where those related data were not part of the initial data. Hence, if there are N users, the application will make 1 query to get all users and then N additional queries to get the articles for each user, totaling N+1 queries.

**N+1 Resolution**

Hibernate & Spring Data JPA provide mechanism to solve the N+1 ORM issue.

At SQL level, what ORM needs to achieve to avoid N+1 is to fire a query that joins the two tables and get the combined results in single query.

**Plain SQL**

**SELECT p.id, p.name FROM Parent p LEFT OUTER JOIN child c ON p.id = c.parent\_id**

**SELECT \* FROM t\_mobile\_vendor vendor LEFT OUTER JOIN t\_phone\_model model ON model.vendor\_id=vendor.vendor\_id**

**1. Spring Data JPA Approach**

If we are using Spring Data JPA, then we have two options to achieve this - using EntityGraph or using select query with fetch join.

public interface UserRepository extends CrudRepository<User, Long> {

List<User> findAllBy();

**@Query("SELECT u FROM User u LEFT JOIN FETCH u.roles")**

List<User> findWithoutNPlusOne();

**@EntityGraph(attributePaths = {"roles"})**

List<User> findAll();

}

**2. Hibernate Approach**

If its pure Hibernate, then the following solutions will work.

**Using HQL Query**

"from User u join fetch u.roles roles roles"

**Using Hibernate Criteria API**

Criteria criteria = session.createCriteria(User.class);

criteria.setFetchMode("roles", FetchMode.EAGER);