HIBERNATE DOCUMENT

Table of Contents

[Case 1: Hibernate Mapping 3](#_Toc59966453)

[Case 2: Hibernate Annotation 3](#_Toc59966454)

[Case 3: Hibernate Embeddable Method 1 4](#_Toc59966455)

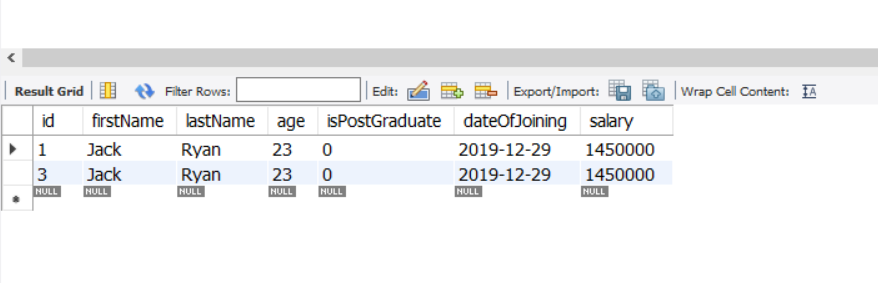
[Case 4: Hibernate Embeddable Method 2 4](#_Toc59966456)

[Case 5: One to One Unidirectional 4](#_Toc59966457)

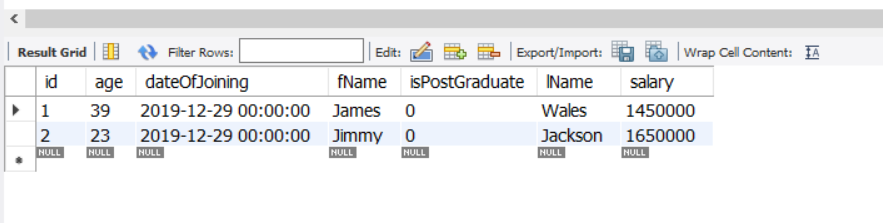
[Case 6: One to One Bidirectional 4](#_Toc59966458)

[Case 7: One to One Unidirectional OR Bidirectional with Shared Primary Key 5](#_Toc59966459)

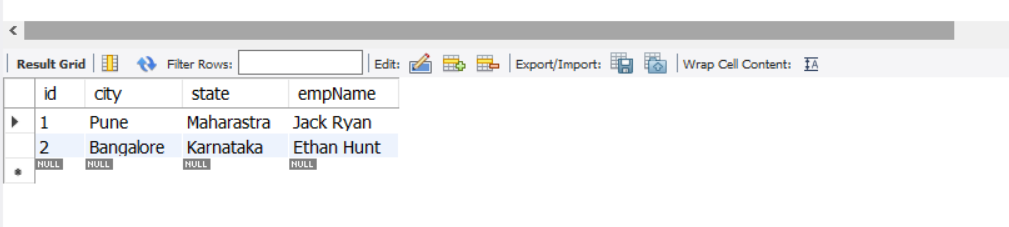
# Case 1: Hibernate Mapping



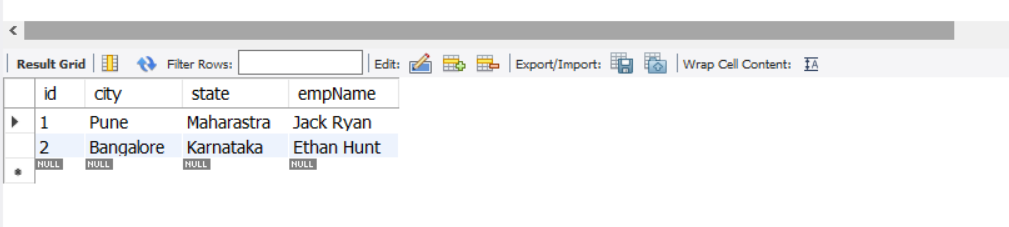
# Case 2: Hibernate Annotation



# Case 3: Hibernate Embeddable Method 1

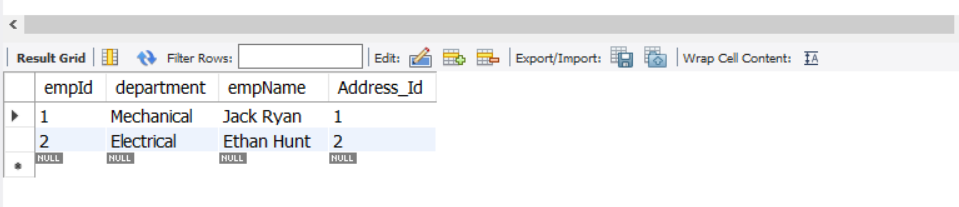


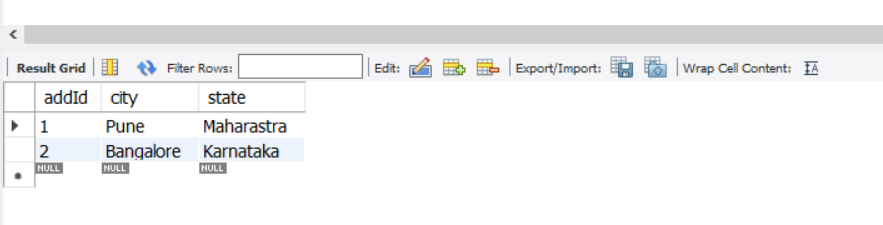
# Case 4: Hibernate Embeddable Method 2



# Case 5: One to One Unidirectional

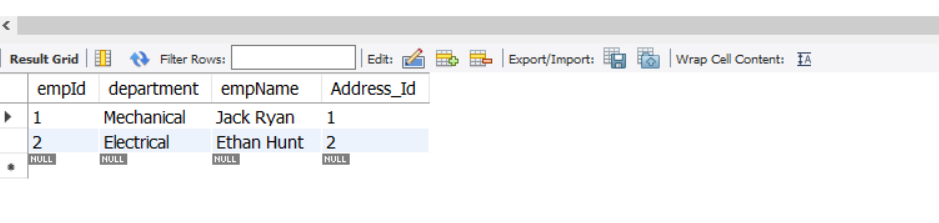
Class Emp contains reference of Address class, but class Address does not contain the reference of Emp class.

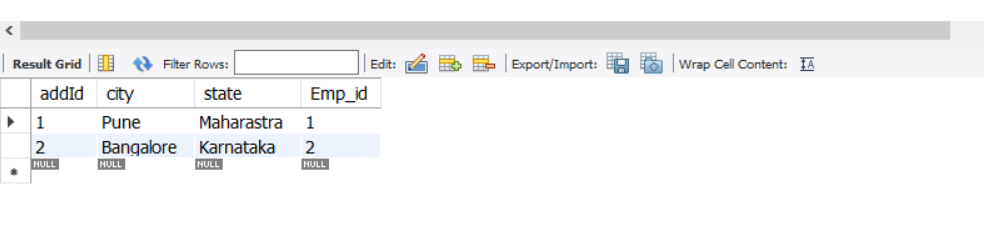




# Case 6: One to One Bidirectional

Class Emp contains reference of Address class, also Address class contains reference of Emp class.

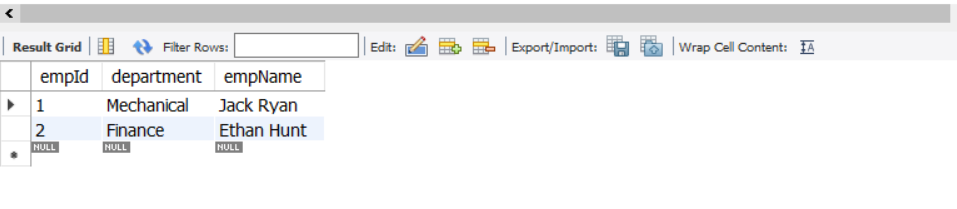


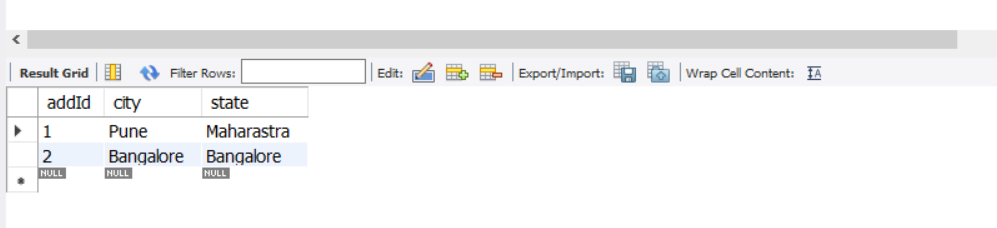


# Case 7: One to One Unidirectional OR Bidirectional with Shared Primary Key

Same Primary Key will be shared among different tables. For example, let’s say Emp1 PK is 1001, then the same PK 1001 will be used in Address table also.

In this case there will be no difference between table structure of unidirectional or bidirectional mapping. Remember, when you use bidirectional mapping @Id annotation should not be used on the table which PK is being set programmatically. Here since we are setting the PK of Address class programmatically, we will not use @Id annotation for addId.





# Case 8: One to Many Join Table

In case of One-to-Many mapping, a separate table is created for mapping the keys.

In this case, an Employee class can have a number of Address class objects. So we create a List of Address objects in Employee class, and in the DB, a separate table mapping the employee with multiple addressId is created.