HIBERNATE DOCUMENT

Table of Contents

[Case 1: Hibernate Mapping 3](#_Toc60005989)

[Case 2: Hibernate Annotation 3](#_Toc60005990)

[Case 3: Hibernate Embeddable Method 1 3](#_Toc60005991)

[Case 4: Hibernate Embeddable Method 2 4](#_Toc60005992)

[Case 5: One to One Unidirectional 4](#_Toc60005993)

[Case 6: One to One Bidirectional 4](#_Toc60005994)

[Case 7: One to One Unidirectional OR Bidirectional with Shared Primary Key 5](#_Toc60005995)

[Case 8: One to Many Unidirectional Join Table 5](#_Toc60005996)

[Case 9: One to Many Unidirectional Join Column 6](#_Toc60005997)

[Case 10: OneToMany Bi-Directional Mapping | JoinColumn v/s MappedBy 6](#_Toc60005998)

[Case 11: OneToMany Bidirectional Mapping Join Table 7](#_Toc60005999)

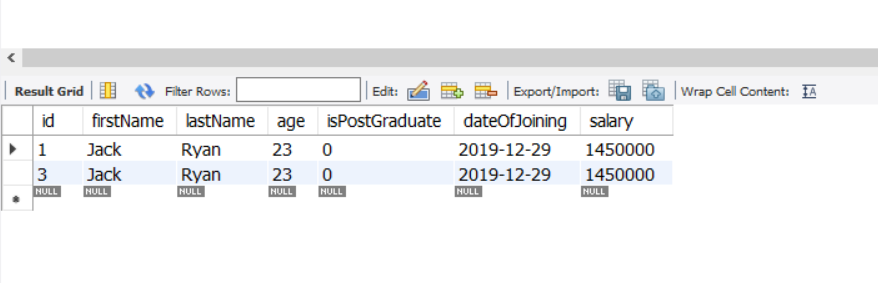
[Case 12: Many To One Unidirectional Join Column 8](#_Toc60006000)

[Case 13: Many To One Unidirectional Join Table 8](#_Toc60006001)

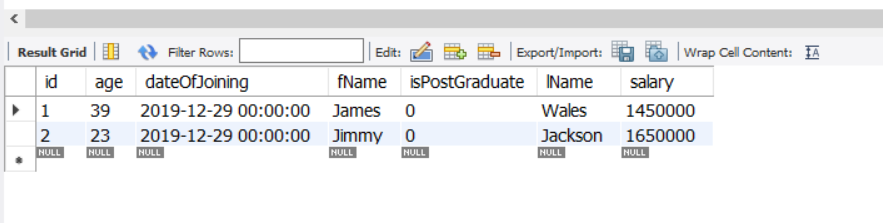
[Case 14: Many To Many with Join Table Unidirectional 9](#_Toc60006002)

[Case 15: Many to Many with Bidirectional 9](#_Toc60006003)

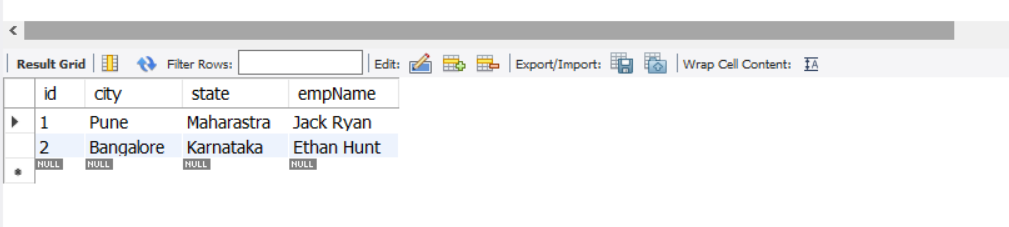
### Case 1: Hibernate Mapping



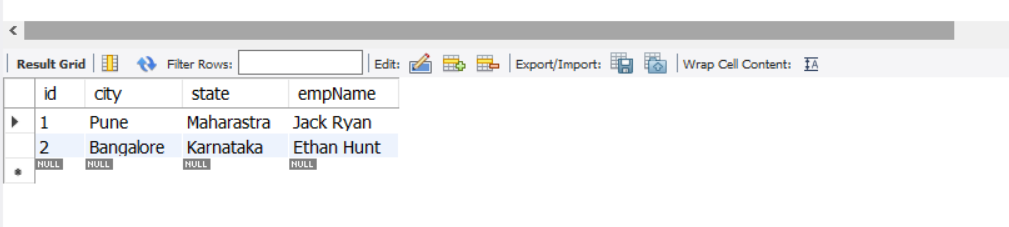
### Case 2: Hibernate Annotation



### Case 3: Hibernate Embeddable Method 1

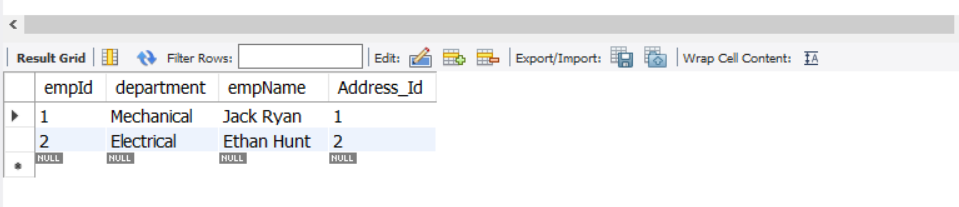


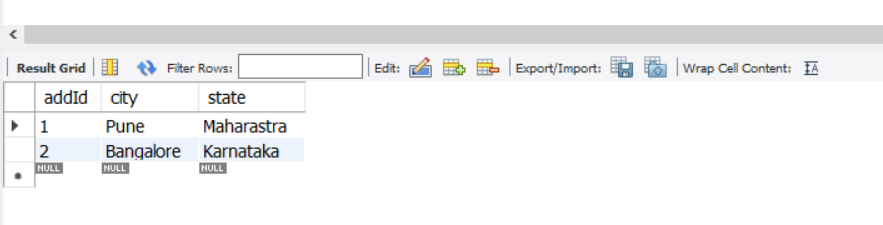
### Case 4: Hibernate Embeddable Method 2



### Case 5: One to One Unidirectional

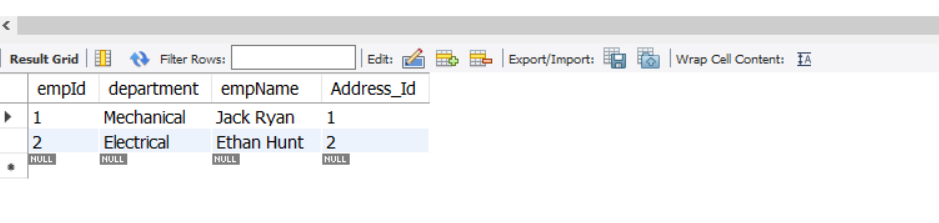
Class Emp contains reference of Address class, but class Address does not contain the reference of Emp class.

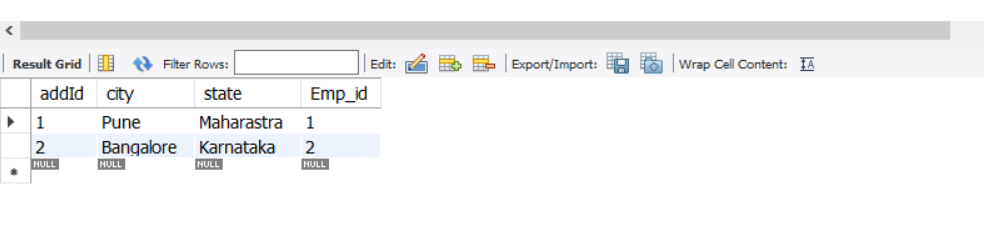




### Case 6: One to One Bidirectional

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

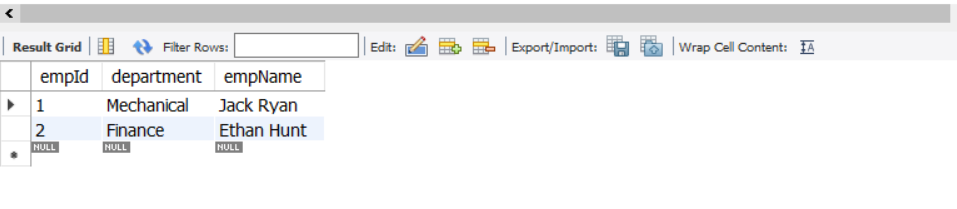


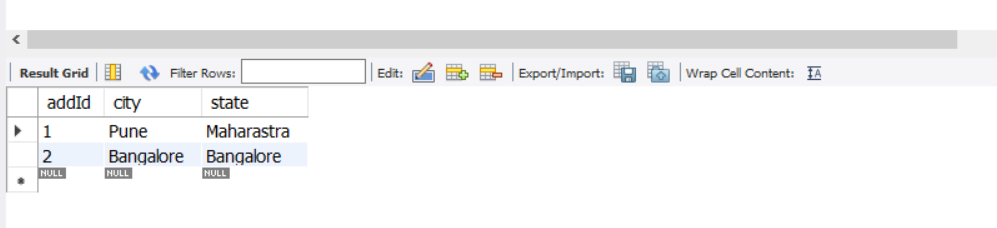


### Case 7: One to One Unidirectional OR Bidirectional with Shared Primary Key

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 Unidirectional Join Table

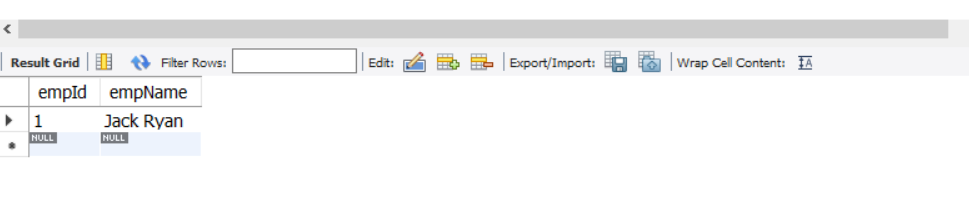
In case of One-to-Many mapping, a separate table is created for mapping the keys.

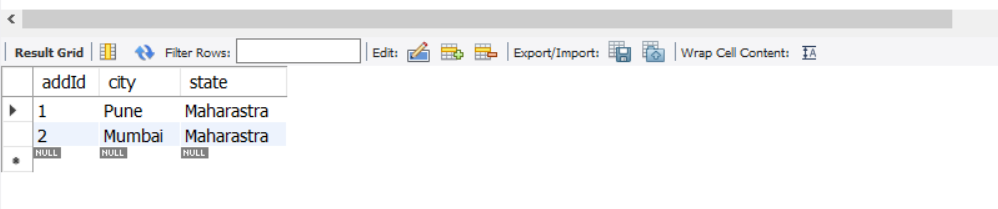
In this case, an Employee class can have a number of Address class objects. So, we create a List of Address objects in Employee class, and in the DB, a separate table mapping the employee with multiple addressId is created.

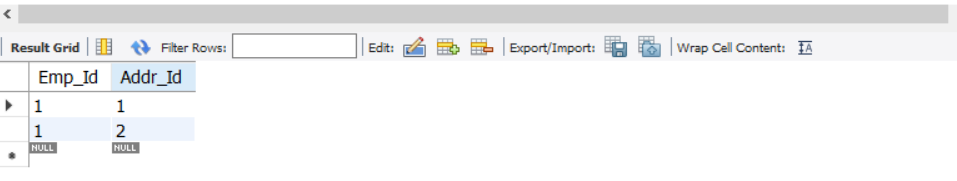
@OneToMany

@JoinTable(name = "Emp\_Addr\_Map",joinColumns = {@JoinColumn(name="Emp\_Id")},

inverseJoinColumns = {@JoinColumn(name="Addr\_Id")})



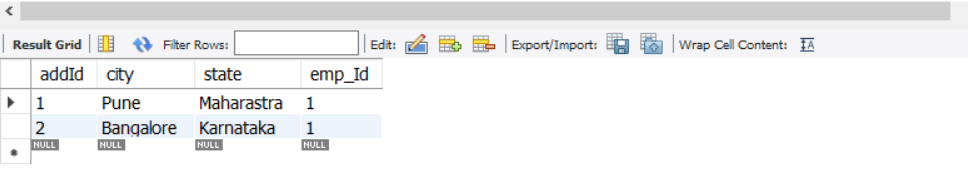




### Case 9: One to Many Unidirectional Join Column

In this case @JoinColumn annotation is used. An extra column emp\_id will be created in Address class table which will hold the Employee class PK.





### Case 10: OneToMany Bi-Directional Mapping | JoinColumn v/s MappedBy

MappedBy and JoinColumn cannot be used simultaneously. Mappedby is only available in OnetoMany and is not available in ManyToOne. Mappedby name should be used in table having OneToMany relationship and mappedby name should be exact as the property name in ManyToOne table.

In Employee9 class:

@OneToMany(mappedBy = "employee9")

**private** List<Address9> address;

In Address9 class:

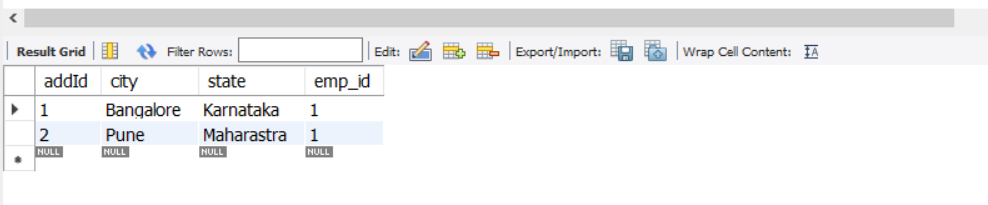
@ManyToOne

@JoinColumn(name="emp\_id")

**private** Employee9 employee9;

\*\* In Employee9 table, mappedby=”employee9” is used. Here “employee9” should be match the name used for Employee9 property in Address9 class.

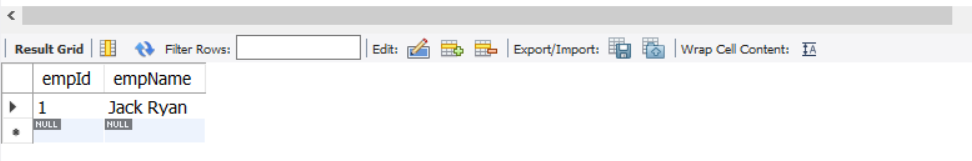


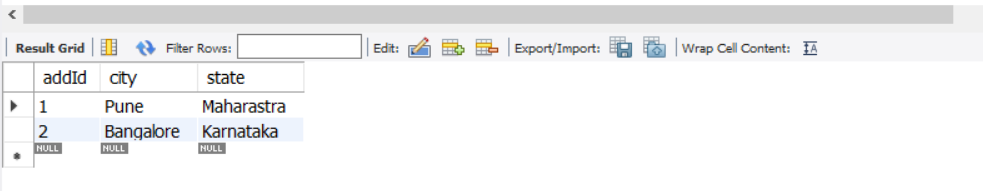


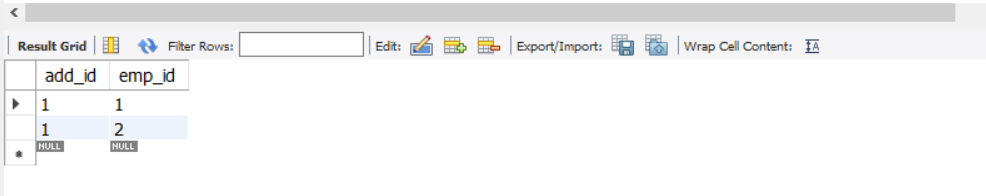
### Case 11: OneToMany Bidirectional Mapping Join Table

In case OneToMany= By default Join Table is used if nothing is specified.

In case of ManyToOne= By default Join Column is used if nothing is specified.



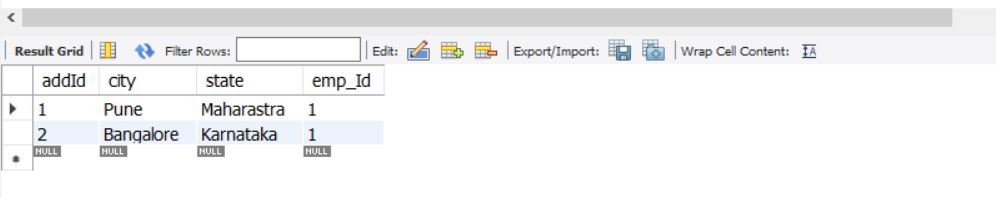




### Case 12: Many To One Unidirectional Join Column

In this case only Employee reference is to be created in Address class.





### Case 13: Many To One Unidirectional Join Table

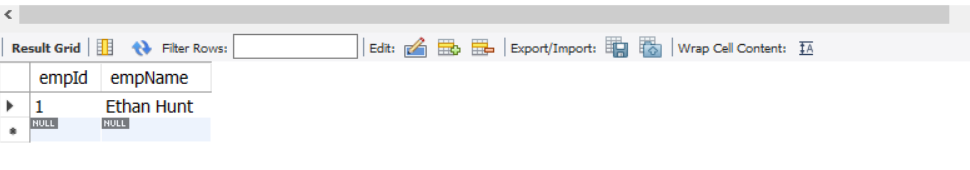
In Address class:

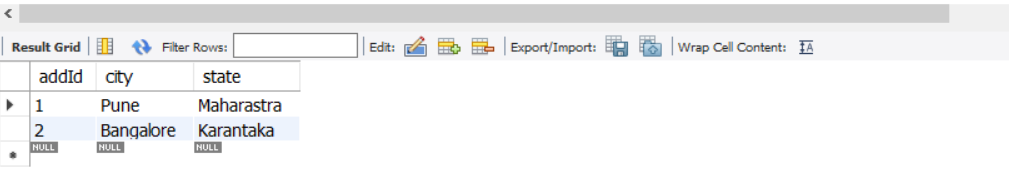
@ManyToOne

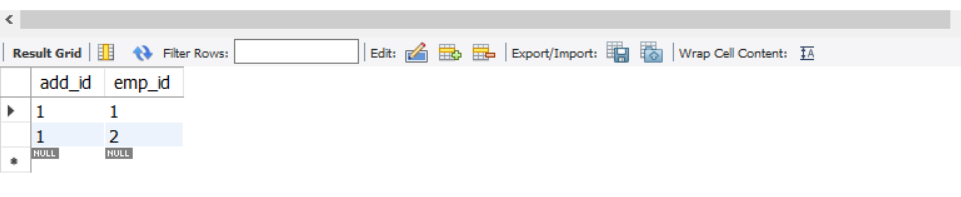
@JoinTable(name="Emp\_Addr",joinColumns = {@JoinColumn(name="emp\_id")},

inverseJoinColumns = {@JoinColumn(name="add\_id")} )

**private** Employee12 employee12;

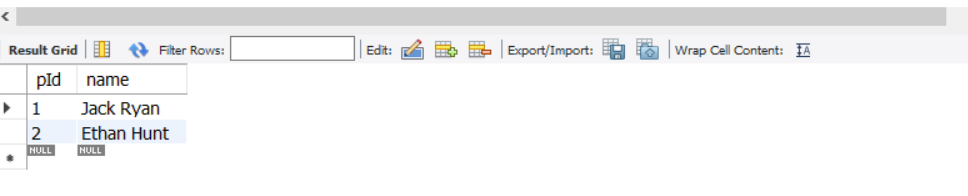




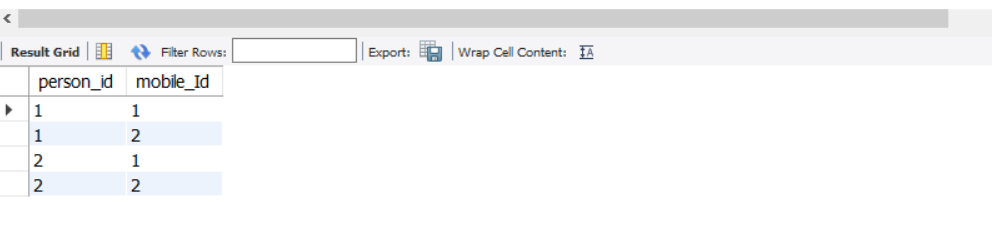


### Case 14: Many To Many with Join Table Unidirectional

Many To Many also uses Join Table by default.







### Case 15: Many to Many with Bidirectional

In this case, mapped is used in the table where join table is not specified.

In Mobile class:

@ManyToMany

@JoinTable(name="Person\_Mobile",joinColumns = {@JoinColumn(name="person\_Id")},

inverseJoinColumns = {@JoinColumn(name="mobile\_id")})

**private** List<Person> persons;

In Person class:

@ManyToMany(mappedBy = "persons")

**private** List<Mobile> mobiles;

