Hibernate

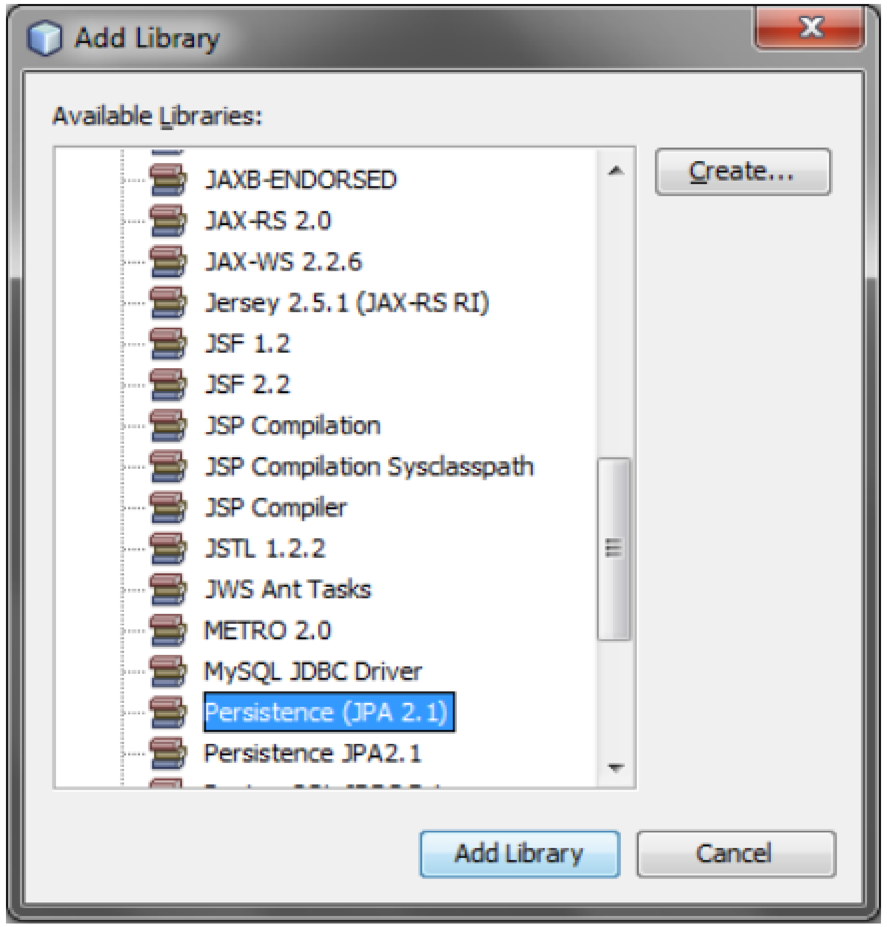
Mapping between tables

- One-to-one mapping
- CascadeType in hibernate
- Unidirectional and bidirectional one-to-one mapping

Relational mapping between DB tables

- Types of Relational Mapping
  - One-to-One
    - E.g., One student has one home address
  - Many-to-One
    - E.g., Many students have same college address
  - One-to-Many
    - E.g., One student has many Phone numbers
  - Many-to-Many
    - E.g., One or more students participate in one or more competitions/exams

Additional Java library needed for use of annotations



Add Library

Available Libraries:

JAXB-ENDORSED
JAX-RS 2.0
JAX-WS 2.2.6
Jersey 2.5.1 (JAX-RS RI)
JSF 1.2
JSF 2.2
JSP Compilation
JSP Compilation Sysclasspath
JSP Compiler
JSTL 1.2.2
JWS Ant Tasks
METRO 2.0
MySQL JDBC Driver
Persistence (JPA 2.1)
Persistence JPA2.1

One-to-one mapping

- Two cases for one-to-one relations
  - Related Entities/Tables share same primary keys values
    - E.g., Student has StudentDetails
    - Both Student and StudentDetails tables have same primary key (Columns)
  - Foreign key is held by one of the Entities/Tables
    - E.g., Student has StudentDetails
    - Do not share common key (column)

One-to-one mapping (key sharing)

- If each row in Table A is linked to a single row in Table B
- The number of rows in Table A = the number of rows in Table B

key sharing

- student

student_id	student_name
1	H B Prajapati
2	Prajapati H B

student_details

student_id	student_address
1	Dept. of IT
2	IT Dept.

- In this example, student_id in student_details table is a foreign key reference to student_id in student table.
- Constraint is present in student_details table
  - student_details is going to accept only those values of student_id which are present in student table.

One-to-one mapping

- First create two classes without any relation established:
  - Student (Entity class)
  - StudentDetails (Entity class)

Student class (Without any annotations for relation)

```
import javax.persistence.CascadeType;
import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.Id;
import javax.persistence.OneToOne;

@Entity
public class Student {
   @Id @GeneratedValue
   private int student_id;
   private String student_name;
   public int getStudent_id() {
      return student_id;
   }
```

Student class (Without any annotations for relation)

```
    public String getStudent_name() {
        return student_name;
    }

    public void setStudent_name(String student_name) {
        this.student_name = student_name;
    }
}
```

StudentDetails class (Without any annotations for relation)

```
@Entity
@Table(name="STUDENT_DETAILS")
public class StudentDeatails {

    @Id @GeneratedValue
    private int student_id;

    private String student_address;

    public int getStudent_id() {
        return student_id;
    }
```

StudentDetails class (Without any annotations for relation)

```
    public String getStudent_address() {
        return student_address;
    }

    public void setStudent_address(String student_address) {
        this.student_address = student_address;
    }

}
```

Configuration file

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE hibernate-configuration PUBLIC "-//Hibernate/Hibernate
    Configuration DTD 3.0//EN" "http://hibernate.sourceforge.net/hibernate-
    configuration-3.0.dtd">
<hibernate-configuration>
 <session-factory>
  <property name="hibernate.dialect">org.hibernate.dialect.MySQLDialect
   </property>
  <property
   name="hibernate.connection.driver_class">com.mysql.jdbc.Driver
   </property>
  <property
   name="hibernate.connection.url">jdbc:mysql://localhost:3306/ddu?zeroD
   ateTimeBehavior=convertToNull </property>
```

Configuration file

```
<property name="hibernate.connection.username">root</property>
<property name="hibernate.hbm2ddl.auto">create</property>
   <mapping class="hibernate.relations.Student"/>
   <mapping class="hibernate.relations.StudentDetails"/>
 </session-factory>
</hibernate-configuration>
```

Establish one-to-one mapping using annotations

- In StudentDetails class, add the following code for establishing relation:

```
@OneToOne(cascade = CascadeType.ALL)
@JoinColumn(name="student_id")
private Student student;
```

- The code indicates that StudentDetails class has one-to-one relationship with Student class on its student_id column.
- Name of the main table (represented by Student entity) is implicit from the following:

```
private Student student;
```

Establish one-to-one mapping using annotations

- In StudentDetails class, add following code for taking value of auto generated primary key value of Main table (Student entity).

```
@Id @GeneratedValue(generator = "newGenerator“)
@GenericGenerator(name="newGenerator", strategy="foreign",
parameters={@Parameter(value="student", name="property")})
private int student_id;
```

- The code indicates that the value of student_id is (foreign key) taken from student_id of student table.
- value="student" tells us from which field the mapping information is available (i.e., defined in private Student student;)

Main class

```
public class Main {
    public static void main(String[] args) {
        Student student=new Student();
        student.setStudent_name("H B Prajapati");
         StudentDetails studentDetails=new StudentDetails();
        studentDetails.setStudent_address("D D University");
        studentDetails.setStudent(student);

        SessionFactory sessionFactory= new
       AnnotationConfiguration().configure().buildSessionFactory();
        Session session=sessionFactory.openSession();
```

Main class

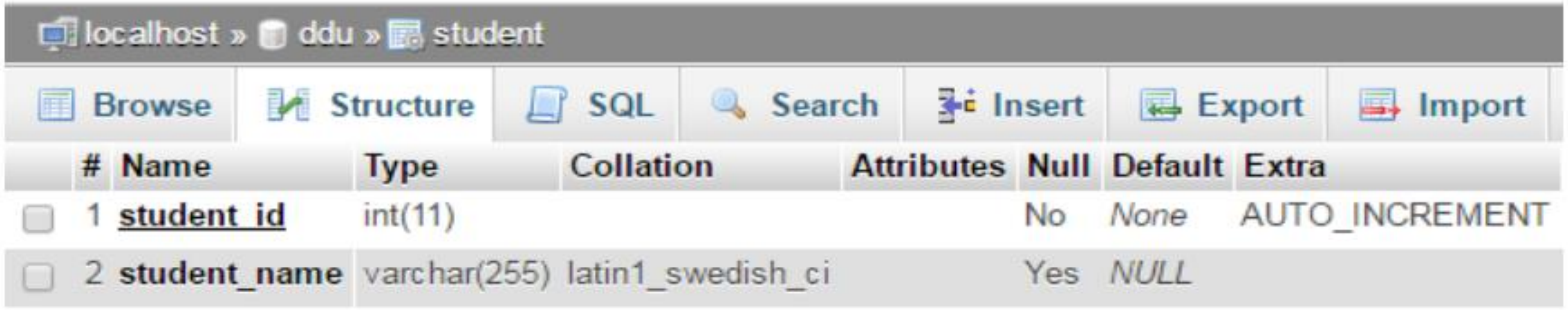
```
        session.beginTransaction();
        //save only studentDetails object
        session.save(studentDetails);

        session.getTransaction().commit();
        session.close();
        sessionFactory.close();


    }

}
```

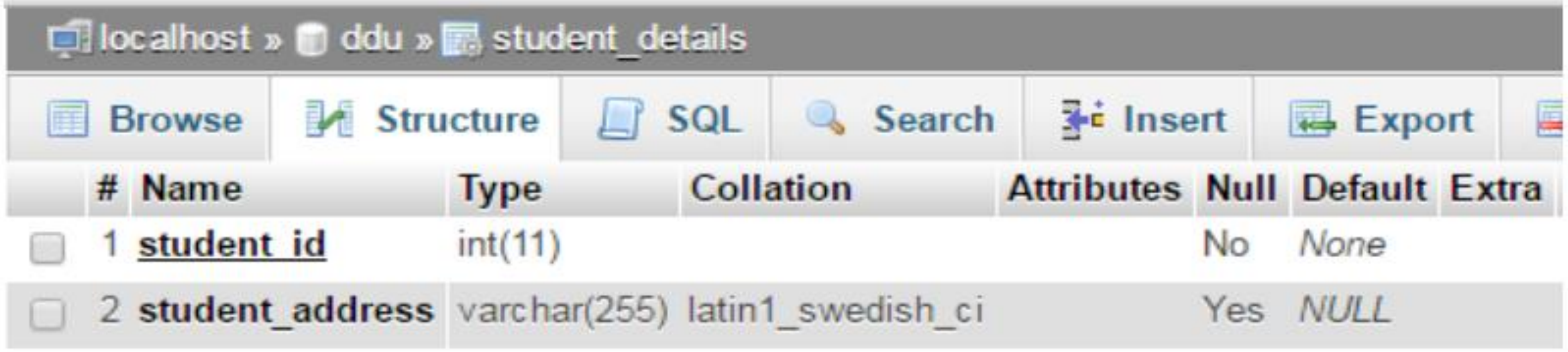
Structure of tables



localhost » ddu » student

Browse | Structure | SQL | Search | Insert | Export | Import

#	Name	Type	Collation	Attributes	Null	Default	Extra
1	student_id	int(11)			No	None	AUTO_INCREMENT
2	student_name	varchar(255)	latin1_swedish_ci		Yes	NULL	

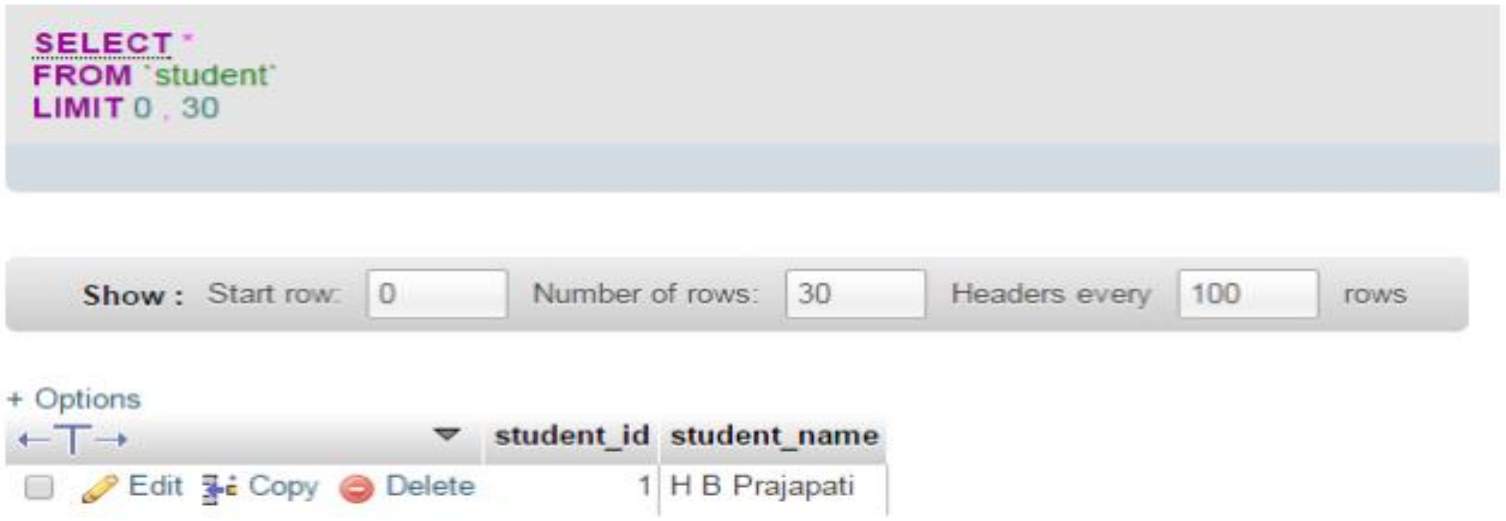


localhost » ddu » student_details

Browse | Structure | SQL | Search | Insert | Export

#	Name	Type	Collation	Attributes	Null	Default	Extra
1	student_id	int(11)			No	None	
2	student_address	varchar(255)	latin1_swedish_ci		Yes	NULL	

Records in the tables

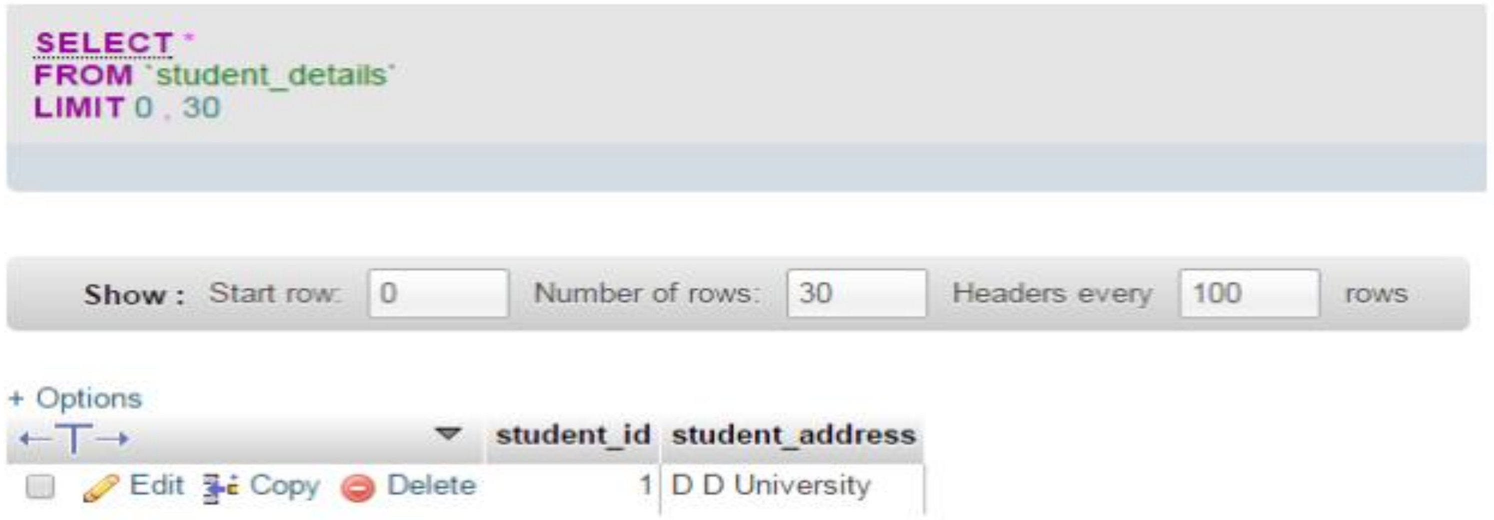


```
SELECT *
FROM `student`
LIMIT 0 , 30
```

Show : Start row: 0 Number of rows: 30 Headers every 100 rows

+ Options

	student_id	student_name
Edit Copy Delete	1	H B Prajapati



```
SELECT *
FROM `student_details`
LIMIT 0 , 30
```

Show : Start row: 0 Number of rows: 30 Headers every 100 rows

+ Options

	student_id	student_address
Edit Copy Delete	1	D D University

Use of one-to-one mapping

- Student (Parent)

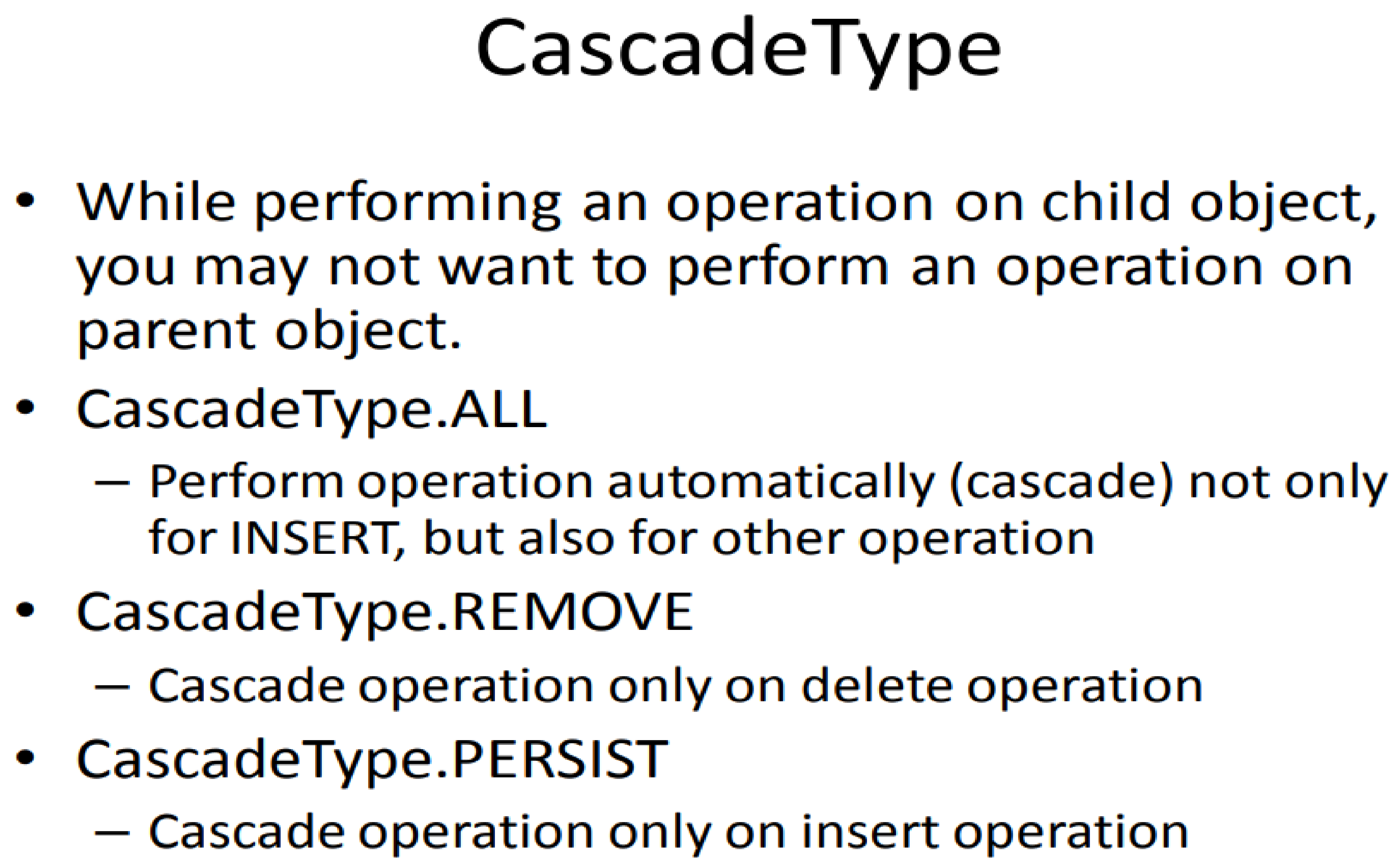
student_id	student_name
1	H B Prajapati
2	Prajapati H B

StudentDetails(Child)

student_id	student_address
1	Dept. of IT
2	IT Dept.

- In Java code, we pass object of parent into child, and then just save child object.
  - Hibernate first writes parent object
  - Hibernate uses same student_id value for child object
  - Performing an operation on a child object also results in performing an operation on the related parent object also
- If we do not establish one-to-one relationship in our java code, then we need to perform each operation ourselves

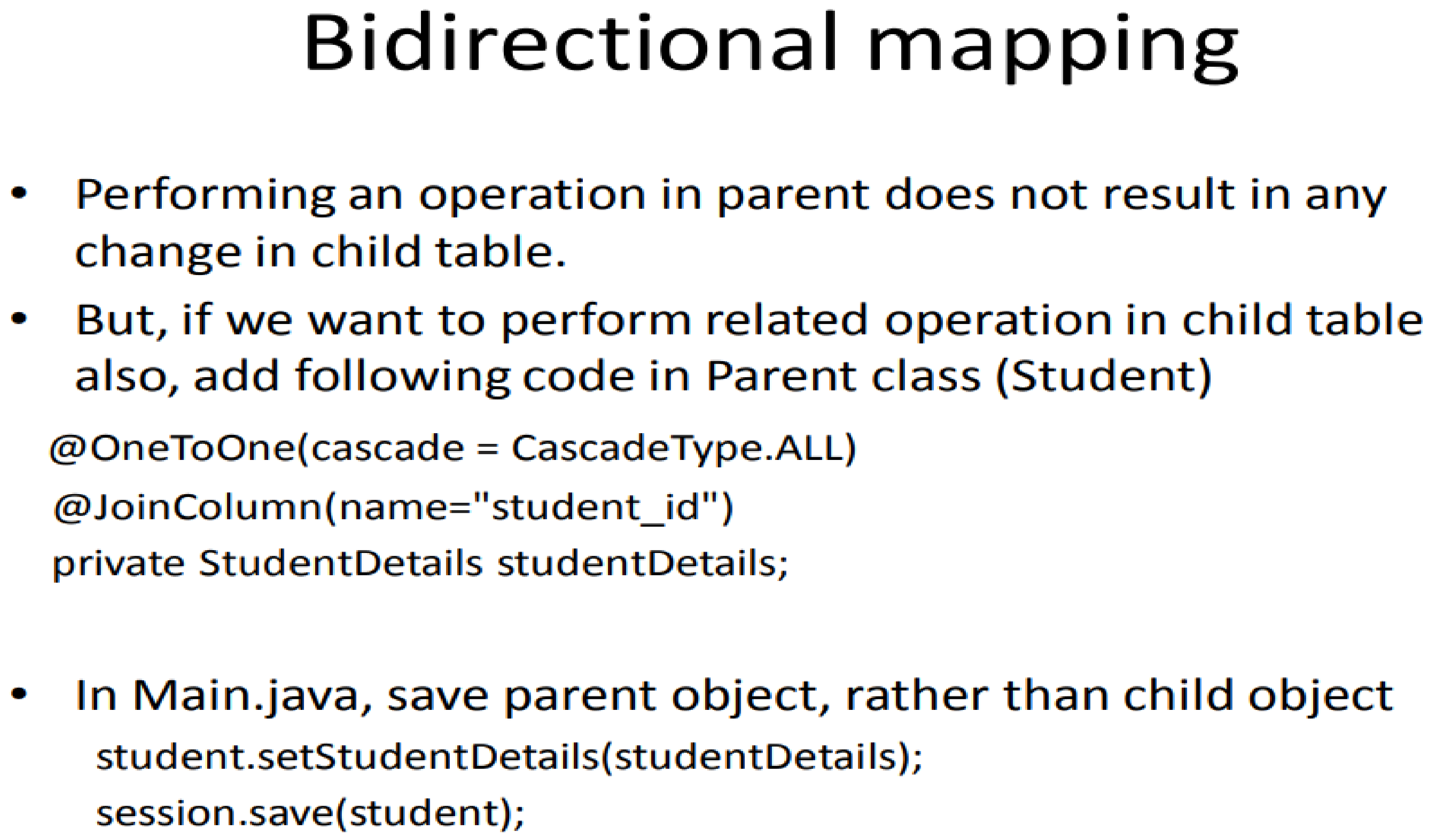
CascadeType is a property used to define cascading in a relationship between a parent and a child.



CascadeType

- While performing an operation on child object, you may not want to perform an operation on parent object.
- CascadeType.ALL
  - Perform operation automatically (cascade) not only for INSERT, but also for other operation
- CascadeType.REMOVE
  - Cascade operation only on delete operation
- CascadeType.PERSIST
  - Cascade operation only on insert operation

Bidirectional association **allows us to fetch details of dependent object from both side**. In such case, we have the reference of two classes in each other.



Bidirectional mapping

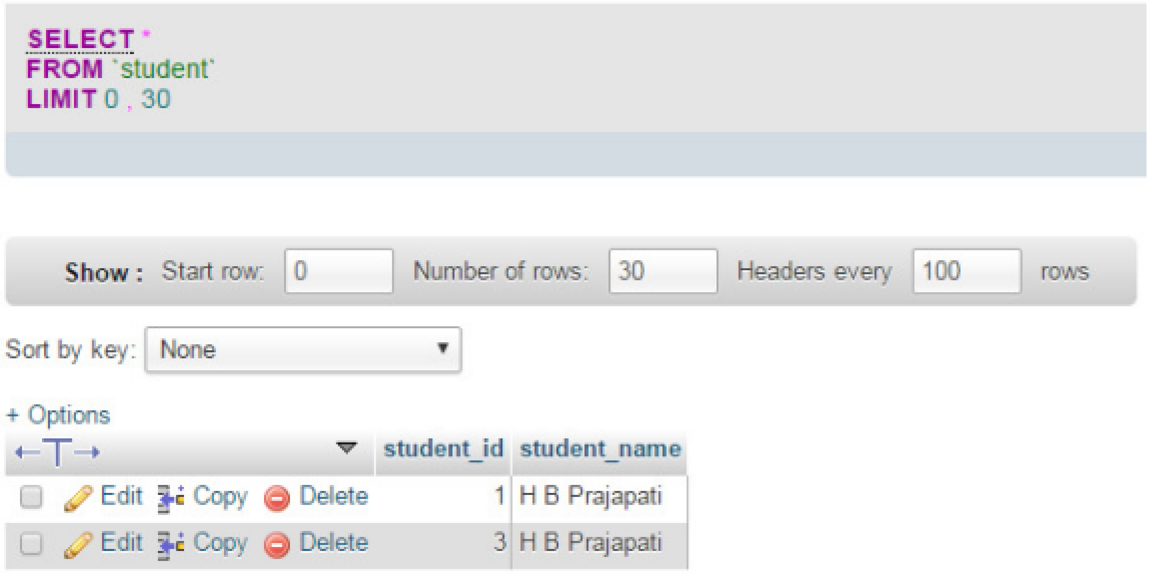
- Performing an operation in parent does not result in any change in child table.
- But, if we want to perform related operation in child table also, add following code in Parent class (Student)

```
@OneToOne(cascade = CascadeType.ALL)
@JoinColumn(name="student_id")
private StudentDetails studentDetails;
```

- In Main.java, save parent object, rather than child object

```
student.setStudentDetails(studentDetails);
session.save(student);
```

Results after execution



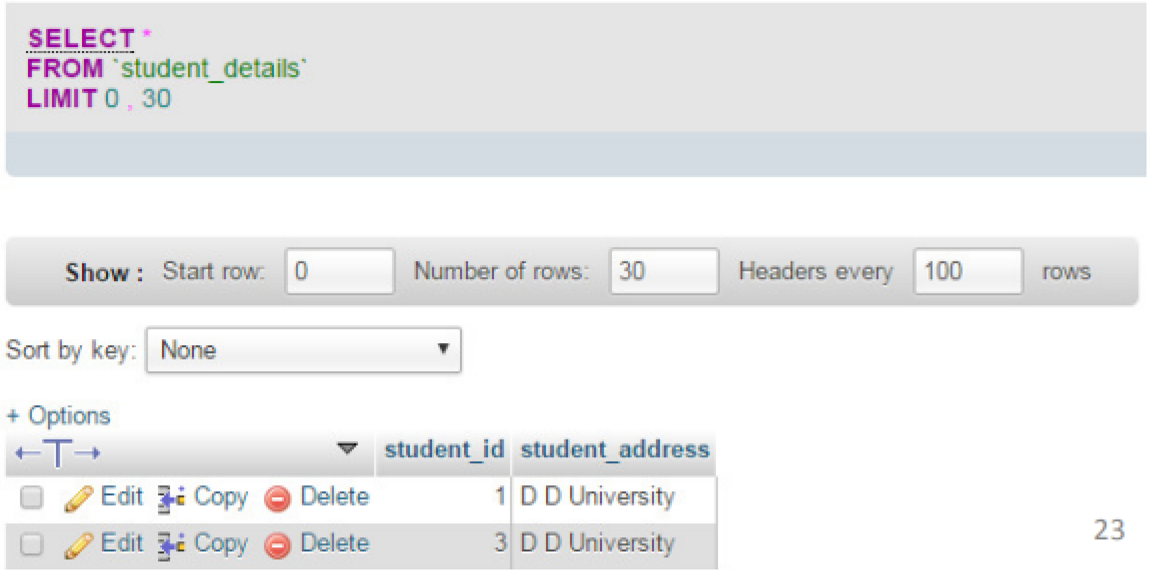
```
SELECT *
FROM `student`
LIMIT 0 , 30
```

Show : Start row: 0 Number of rows: 30 Headers every 100 rows

Sort by key: None

+ Options

	student_id	student_name
Edit Copy Delete	1	H B Prajapati
Edit Copy Delete	3	H B Prajapati



```
SELECT *
FROM `student_details`
LIMIT 0 , 30
```

Show : Start row: 0 Number of rows: 30 Headers every 100 rows

Sort by key: None

+ Options

	student_id	student_address
Edit Copy Delete	1	D D University
Edit Copy Delete	3	D D University

One-to-one mapping (separate key in child)

- If each row in Table A is linked to a single row in Table B
- The number of rows in Table A = the number of rows in Table B

student

student_id	student_name	address_id
1	H B Prajapati	1
2	Prajapati H B	2

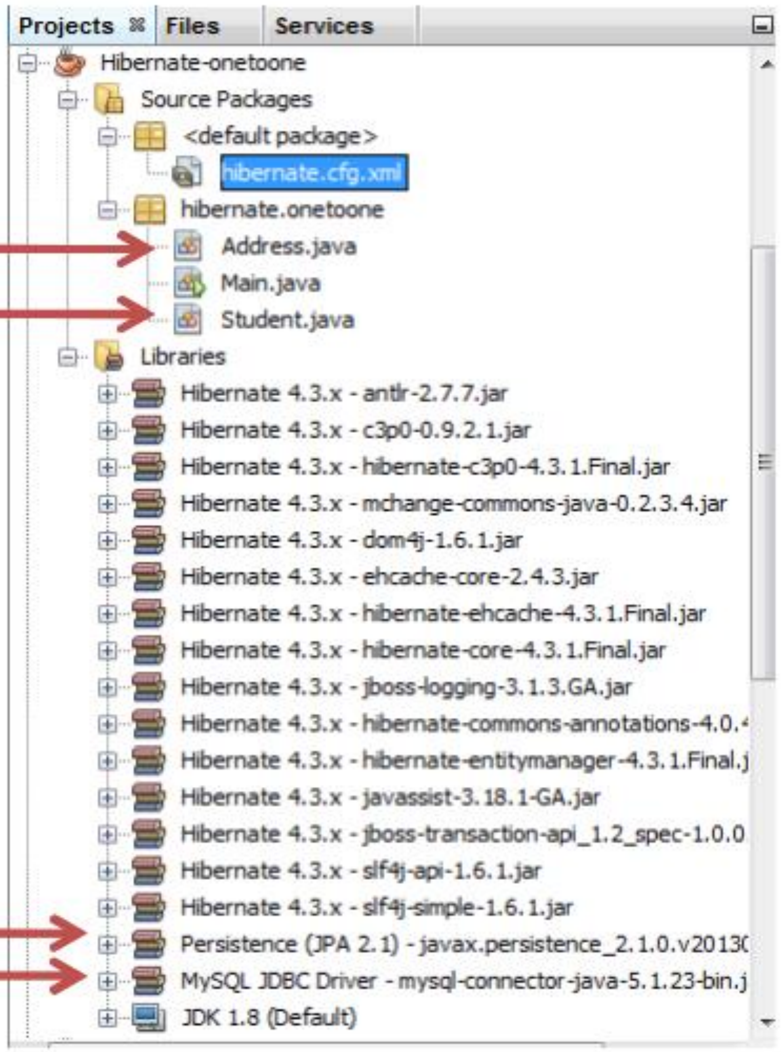
address

address_id	...
1	...
2	...

Key of student table

Key of address table

One-to-one mapping



```
Hibernate-onetoone
  Source Packages
    <default package>
      hibernate.cfg.xml
    hibernate.onetoone
      Address.java
      Main.java
      Student.java
  Libraries
    Hibernate 4.3.x - antlr-2.7.7.jar
    Hibernate 4.3.x - c3p0-0.9.2.1.jar
    Hibernate 4.3.x - hibernate-c3p0-4.3.1.Final.jar
    Hibernate 4.3.x - mchange-commons-java-0.2.3.4.jar
    Hibernate 4.3.x - dom4j-1.6.1.jar
    Hibernate 4.3.x - ehcache-core-2.4.3.jar
    Hibernate 4.3.x - hibernate-ehcache-4.3.1.Final.jar
    Hibernate 4.3.x - hibernate-core-4.3.1.Final.jar
    Hibernate 4.3.x - jboss-logging-3.1.3.GA.jar
    Hibernate 4.3.x - hibernate-commons-annotations-4.0.
    Hibernate 4.3.x - hibernate-entitymanager-4.3.1.Final.j
    Hibernate 4.3.x - javassist-3.18.1-GA.jar
    Hibernate 4.3.x - jboss-transaction-api_1.2_spec-1.0.0
    Hibernate 4.3.x - slf4j-api-1.6.1.jar
    Hibernate 4.3.x - slf4j-simple-1.6.1.jar
    Persistence (JPA 2.1) - javax.persistence_2.1.0.v2013
    MySQL JDBC Driver - mysql-connector-java-5.1.23-bin.j
    JDK 1.8 (Default)
```

Entity classes

Libraries

Student class

```
package hibernate.onetoone;
import javax.persistence.CascadeType;
import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.Id;
import javax.persistence.OneToOne;

@Entity
public class Student {
  @Id @GeneratedValue
  private int student_id;
  public int getStudent_id() {
    return student_id;
  }
```

Student class

```
 private String student_name;
 public String getStudent_name() {
   return student_name;
 }
 public void setStudent_name(String student_name) {
   this.student_name = student_name;
 }
 @OneToOne(cascade = CascadeType.ALL)
 private Address address;
 public Address getAddress() {
   return address;
 }
 public void setAddress(Address address) {
   this.address = address;
 }
}
```

Address class

```
@Entity
public class Address {
    @Id @GeneratedValue
    private int address_id;
    private String street;
    private String city;
    private String state;
    private String pincode;
…
//appropriate getter/setter methods
```

Configuration file

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE hibernate-configuration PUBLIC "-//Hibernate/Hibernate
   Configuration DTD 3.0//EN" "http://hibernate.sourceforge.net/hibernate-
   configuration-3.0.dtd">
<hibernate-configuration>
 <session-factory>
  <property
   name="hibernate.dialect">org.hibernate.dialect.MySQLDialect</property
   >
  <property
   name="hibernate.connection.driver_class">com.mysql.jdbc.Driver</prope
   rty>
  <property
   name="hibernate.connection.url">jdbc:mysql://localhost:3306/ddu?zeroD
   ateTimeBehavior=convertToNull</property>
  <property name="hibernate.connection.username">root</property>
```

Configuration file

```
 <property name="hibernate.hbm2ddl.auto">create</property>
    <mapping class="hibernate.onetoone.Student"/>
    <mapping class="hibernate.onetoone.Address"/>
 </session-factory>
</hibernate-configuration>
```

Main class

```
package hibernate.onetoone;

import org.hibernate.Session;
import org.hibernate.SessionFactory;
import org.hibernate.cfg.AnnotationConfiguration;

public class Main {
    public static void main(String[] args) {
        Student student=new Student();
        student.setStudent_name("H B Prajapati");
```

Main class

```
Address address=new Address();
address.setStreet("I. G. Road");
address.setCity("Nadiad");
address.setState("Gujarat");
address.setPincode("387002");

student.setAddress(address);
```

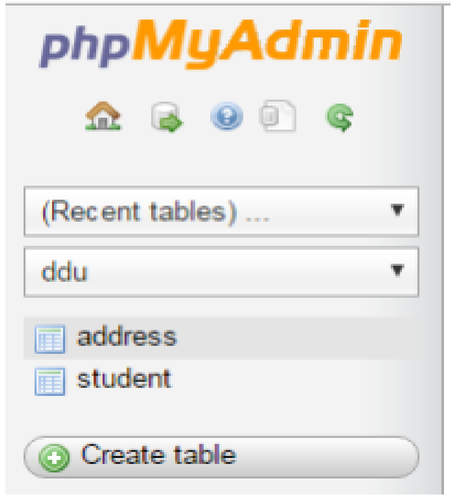
Main class

```
SessionFactory sessionFactory= new
AnnotationConfiguration().configure().buildSessionFactory();
 Session session=sessionFactory.openSession();
 session.beginTransaction();

 //session.save(studentDetails);
 session.save(student);

 session.getTransaction().commit();
 session.close();
 sessionFactory.close();
}
}
```

Results after execution

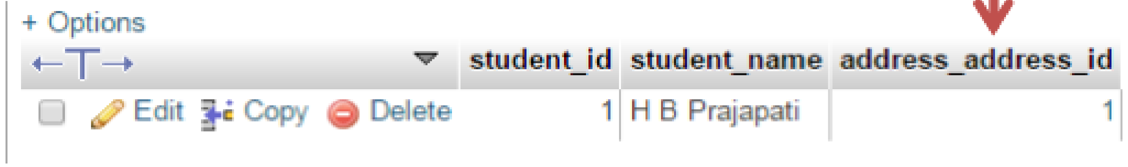


Due to the following code

```
 @OneToOne(cascade = CascadeType.ALL)
   private Address address;
```

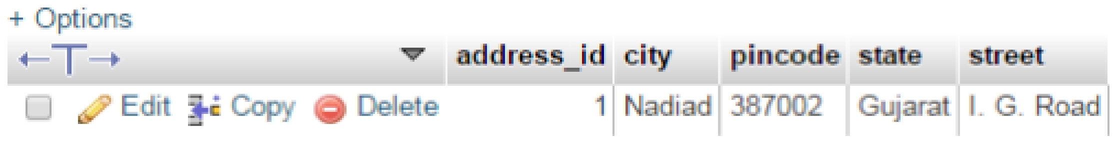
Column automatically added
In column name address_address_id,
address is the name of the field of type Address and address_id is the Id (column) of Address

- Table: student



student_id	student_name	address_address_id
1	H B Prajapati	1

- Table: address



address_id	city	pincode	state	street
1	Nadiad	387002	Gujarat	I. G. Road