HIBERNATE

- create table ddu.STUDENT3 (id INT NOT NULL auto_increment, first_name VARCHAR(20) default NULL, last_name VARCHAR(20) default NULL, semester INT default NULL, PRIMARY KEY (id));

- select * from ddu.student2;

Annotations, hbm2ddl, primary key auto generation

Hibernate application with annotations

What are hibernate annotations?

- Hibernate annotations are the newest way to define mappings between Java class and database table.
- Annotations do not require use of Mapping XML file.
- It uses the following jar files: –

    Persistence JPA API

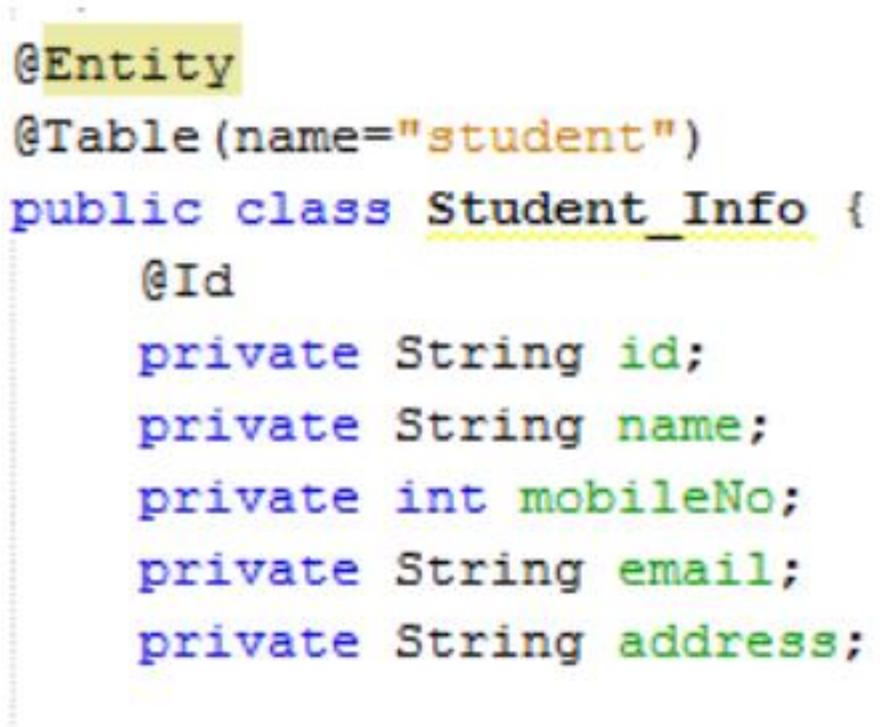
    Hibernate commons annotations

Annotations for Hibernate Entity class

- Two mandatory annotations
  - @Entity
    - To indicate that class is a Hibernate entity
    - Class must have zero argument constructor
  - @Id
    - To indicate that data field is the primary key
- Other annotations
  - @Table (Want to use other name of the table)
  - @GeneratedValue (two parameters: strategy and generator) to generate Id value automatically

Student_Info (Entity class)

- Create data members



```
@Entity
@Table(name="student")
public class Student_Info {
    @Id
    private String id;
    private String name;
    private int mobileNo;
    private String email;
    private String address;
```

- Add getter and setter methods

```
public class Main {
    public static void main(String[] args) {
        Student_Info student = new Student_Info();
        student.setId("2034567");
        student.setName("Fatema");
        student.setEmail("fatemavhora.it@ddu.ac.in");
        student.setMobileNo(986754320);
        student.setAddress("DDU,Nadiad");

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

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

Hibernate Configuration file

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE hibernate-configuration PUBLIC "-//Hibernate/Hibernate
Configuration DTD 3.0//EN" "http://hibernate.sourceforge.net/hibernateconfiguration-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>
<property name="hibernate.connection.username">root</property>
<property name="hibernate.hbm2ddl.auto">create</property>
<mapping class="hibernate.pkg1.Student_Info"/>
</session-factory>
</hibernate-configuration>
```

hbm2ddl

- hbm2ddl Configuration means **hibernate mapping to create schema DDL (Data Definition Language)**.
- Automatically validates or exports schema DDL to the database when the SessionFactory is created.
- With create-drop, the database schema will be dropped when the SessionFactory is closed explicitly.

Property: hibernate.hbm2ddl.auto

- Four possible values
  - create
  - update
  - create-drop
  - validate

hibernate.hbm2ddl.auto=create

- How to test this property?
  - Drop the table
  - Run the program using
    - <property name="hibernate.hbm2ddl.auto">create</property>
  - Table gets created automatically
  - Table also contains inserted record

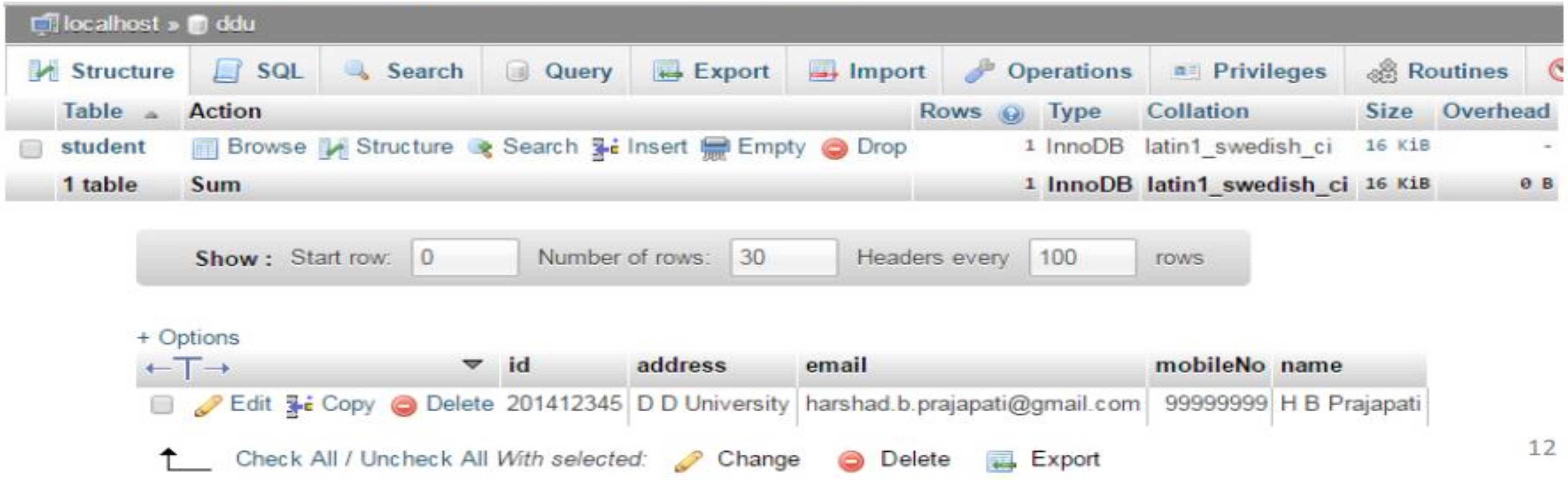


Table	Action	Rows	Type	Collation	Size	Overhead
student	Browse Structure Search Insert Empty Drop	1	InnoDB	latin1_swedish_ci	16 KiB	-
1 table	Sum	1	InnoDB	latin1_swedish_ci	16 KiB	0 B

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

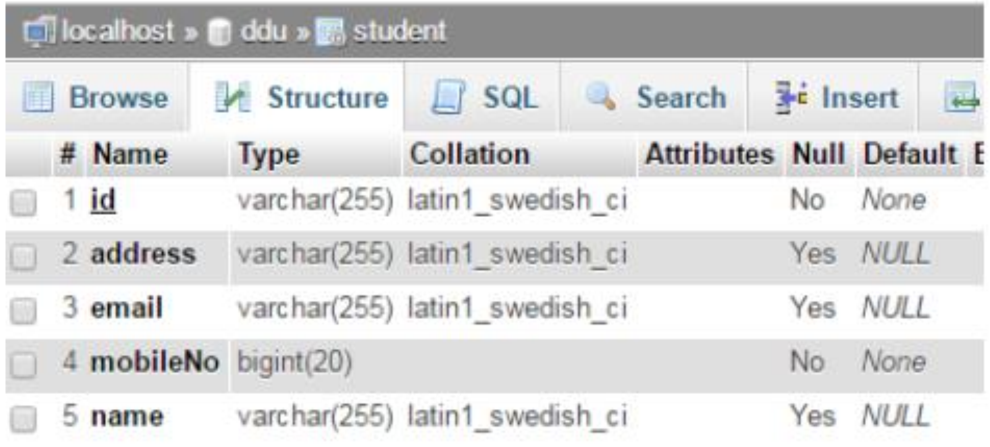
+ Options

	id	address	email	mobileNo	name
Edit Copy Delete	201412345	D D University	harshad.b.prajapati@gmail.com	99999999	H B Prajapati

Check All / Uncheck All *With selected:* Change Delete Export

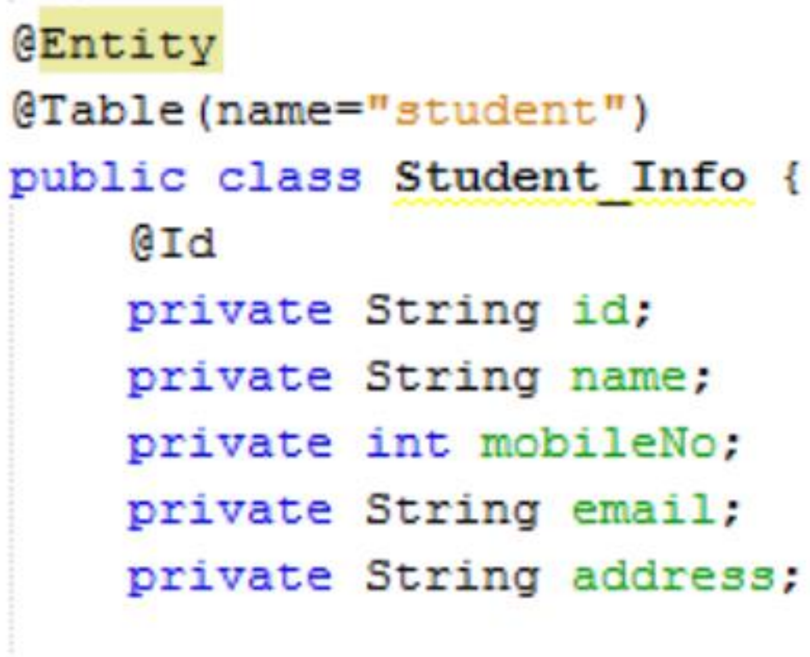
Structure of auto created table

Structure of auto created table:



#	Name	Type	Collation	Attributes	Null	Default
1	id	varchar(255)	latin1_swedish_ci		No	None
2	address	varchar(255)	latin1_swedish_ci		Yes	NULL
3	email	varchar(255)	latin1_swedish_ci		Yes	NULL
4	mobileNo	bigint(20)			No	None
5	name	varchar(255)	latin1_swedish_ci		Yes	NULL

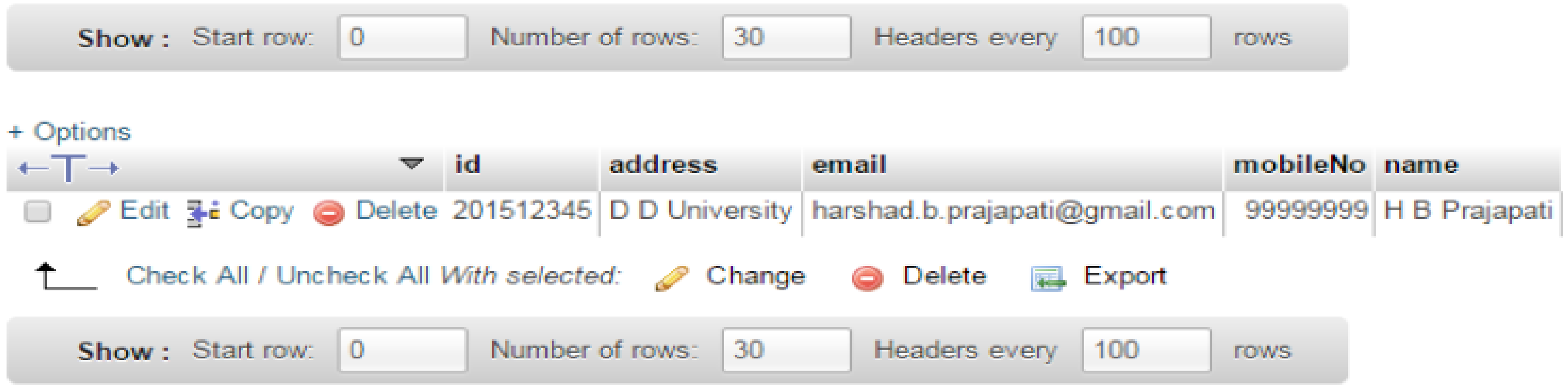
Structure of Model class



```
@Entity
@Table(name="student")
public class Student_Info {
    @Id
    private String id;
    private String name;
    private int mobileNo;
    private String email;
    private String address;
```

What happens to an existing table?

- How to test it?
  - Set the property value
    - <property name="hibernate.hbm2ddl.auto">create</property>
  - Keep the record in the table
  - Write a new record using program
    - student.setId("201512345");
  - Run the program



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

+ Options

	id	address	email	mobileNo	name
Edit Copy Delete	201512345	D D University	harshad.b.prajapati@gmail.com	99999999	H B Prajapati

Check All / Uncheck All *With selected:* Change Delete Export

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

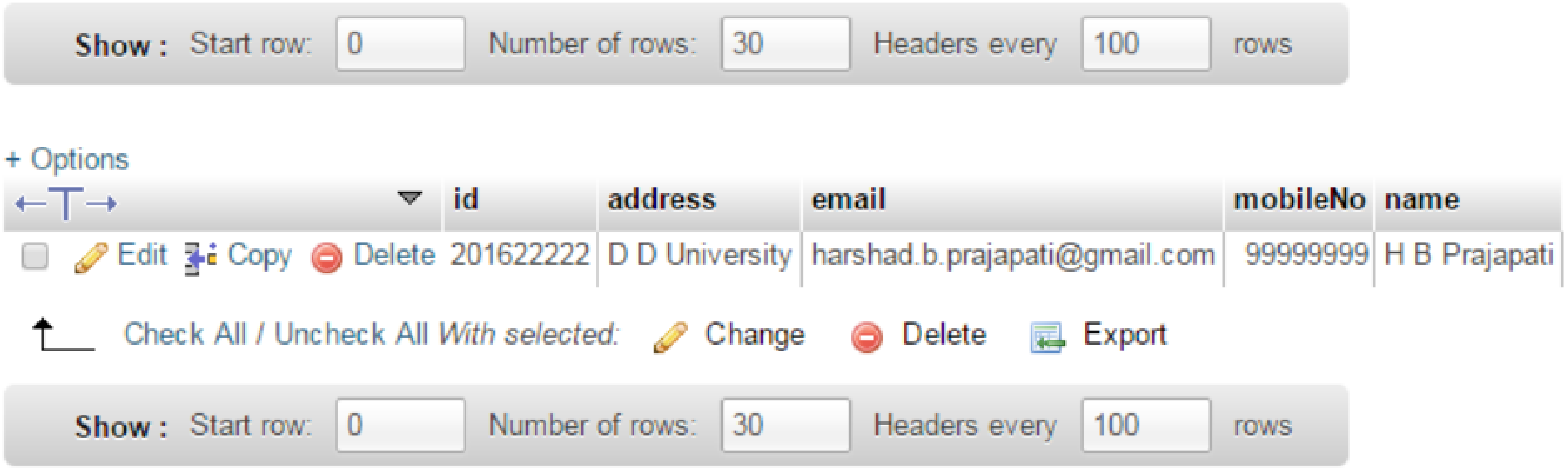
- Using create value: existing table is dropped and a new is created

Create-drop

- Similar to create
- In addition, if session factory is explicitly closed, hibernate drops the table.
- If session factory is not explicitly closed, hibernate works as create, and does not drop the table.

  – `<property name="hibernate.hbm2ddl.auto">create-drop</property>`

- How to test it?
  - In hibernate.cfg.xml file
    - <property name="hibernate.hbm2ddl.auto">create-drop</property>
  - In Main.java
    - student.setId("201622222");
    - // do not close session factory



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

+ Options

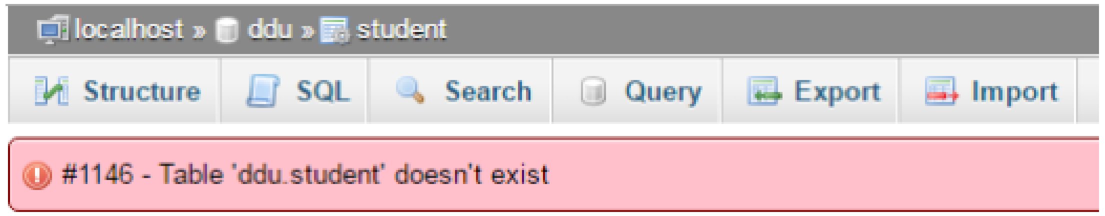
	id	address	email	mobileNo	name
Edit Copy Delete	201622222	D D University	harshad.b.prajapati@gmail.com	99999999	H B Prajapati

Check All / Uncheck All *With selected:* Change Delete Export

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

hibernate.hbm2ddl.auto=create-drop (session factory is closed)

- How to test it?
  - In hibernate.cfg.xml file
    - <property name="hibernate.hbm2ddl.auto">create-drop</property>
  - In Main.java
    - student.setId("201622222");
    - …
    - sessionFactory.close();



#1146 - Table 'ddu.student' doesn't exist

Update

hibernate.hbm2ddl.auto=update

- Every time an application is run, hibernate just updates the schema (i.e., addition of column, change in the column name, but data remains same)
- Some times weird results are observed
- Hibernates has provided it for experimental purpose (should not be used in production code).
- In hibernate.cfg.xml file
  - `<property name="hibernate.hbm2ddl.auto">update</property>`

Validate

hibernate.hbm2ddl.auto=validate

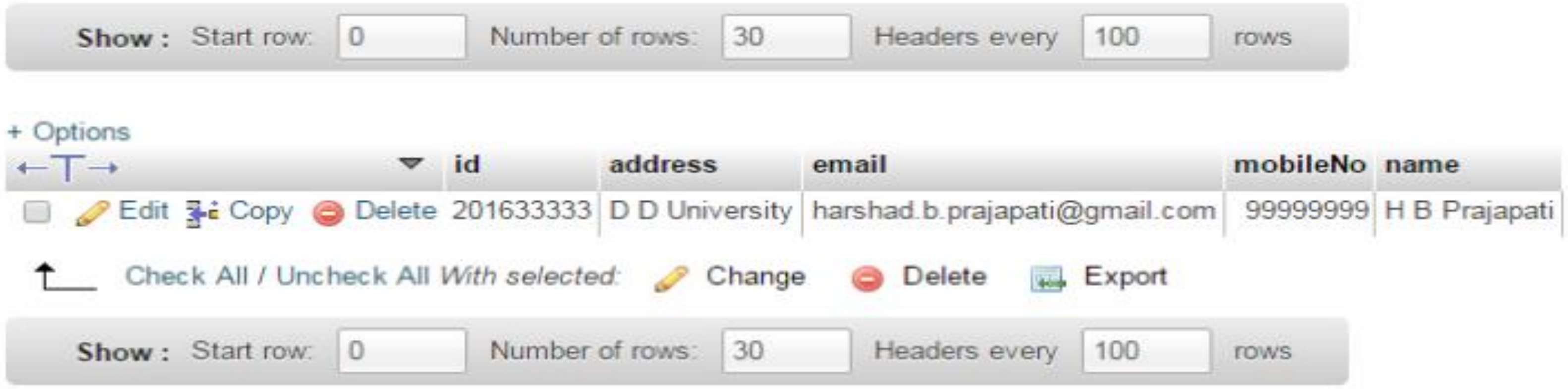
- Hibernate validates with existing schema
- Will not update any data or any change in schema
- In hibernate.cfg.xml file
  - <property name="hibernate.hbm2ddl.auto">validate</property>

Use of values of hbm2dll.auto

- Hibernate says nothing about possible uses of these values
- Do not use in production environment
- Write your own queries to do any modification in schema of tables.
- Do not set this value in production code

Without use of hibernate.hbm2ddl.auto

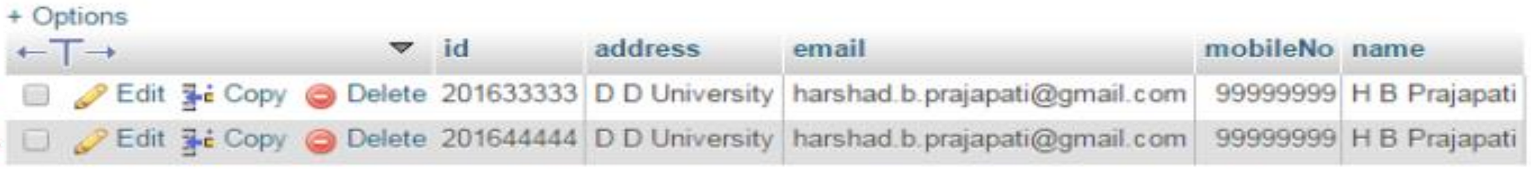
- Run 1: Keep the property (hibernate.hbm2ddl.auto)
- student.setId("201633333");


Show : Start row: 0 Number of rows: 30 Headers every 100 rows
+ Options
Check All / Uncheck All With selected: Change Delete Export
Show : Start row: 0 Number of rows: 30 Headers every 100 rows

	id	address	email	mobileNo	name
Edit Copy Delete	201633333	D D University	harshad.b.prajapati@gmail.com	99999999	H B Prajapati

- Run 2: With remove the property (hibernate.hbm2ddl.auto)
- student.setId("201644444");

From Run 1
From Run 2

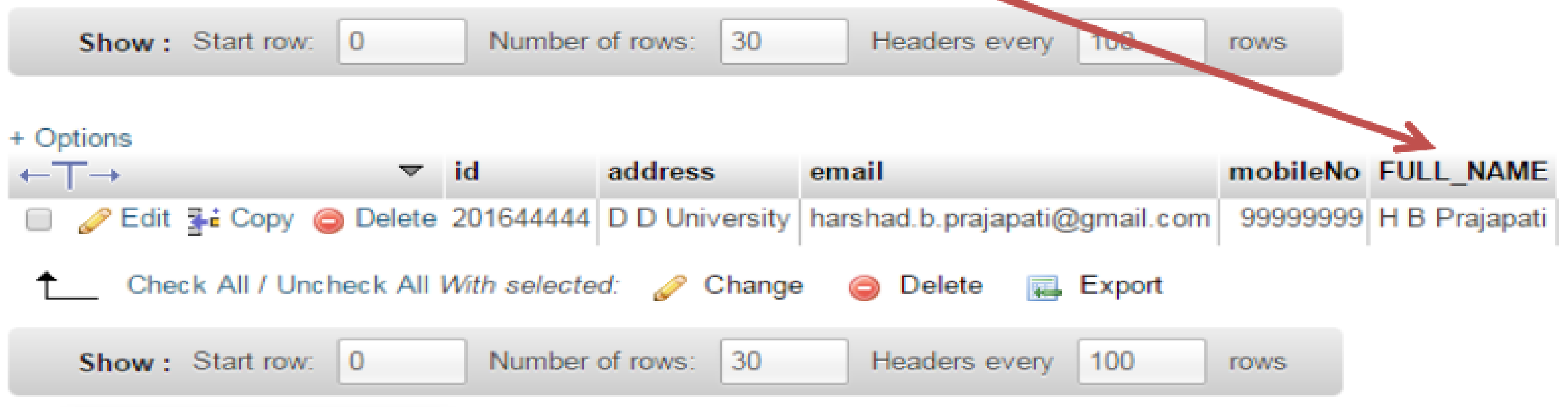


	id	address	email	mobileNo	name
Edit Copy Delete	201633333	D D University	harshad.b.prajapati@gmail.com	99999999	H B Prajapati
Edit Copy Delete	201644444	D D University	harshad.b.prajapati@gmail.com	99999999	H B Prajapati

Annotations for additional configurations

- To use other name for database columns
  - @Column
- In Student_Info.java, do the following change

```
@Column(name="FULL_NAME")
private String name;
```



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

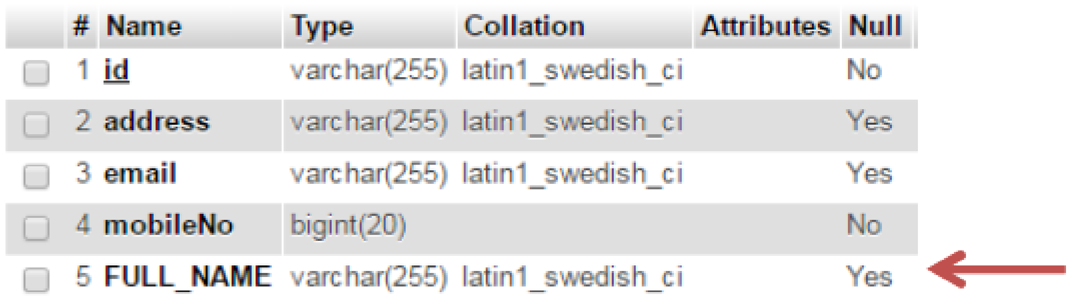
+ Options

	id	address	email	mobileNo	FULL_NAME
Edit Copy Delete	201644444	D D University	harshad.b.prajapati@gmail.com	99999999	H B Prajapati

Check All / Uncheck All *With selected:* Change Delete Export

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

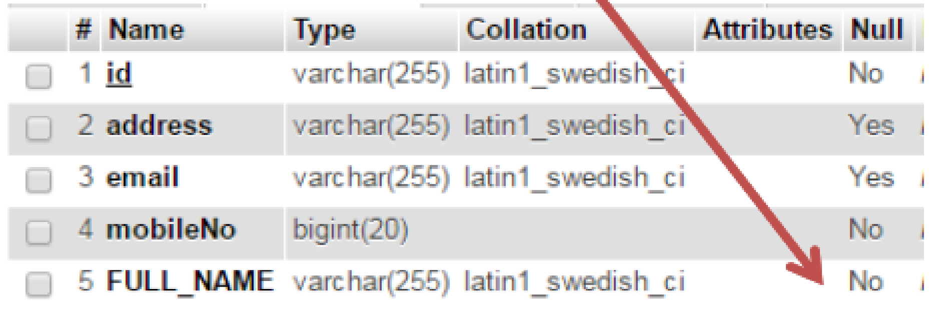
Annotations for additional configurations



	#	Name	Type	Collation	Attributes	Null
☐	1	id	varchar(255)	latin1_swedish_ci		No
☐	2	address	varchar(255)	latin1_swedish_ci		Yes
☐	3	email	varchar(255)	latin1_swedish_ci		Yes
☐	4	mobileNo	bigint(20)			No
☐	5	FULL_NAME	varchar(255)	latin1_swedish_ci		Yes

- We want database column as NOT NULL
  - Use @Column annotation with nullable property
- In Student_Info.java

```
@Column(name="FULL_NAME", nullable=false)
private String name;
```

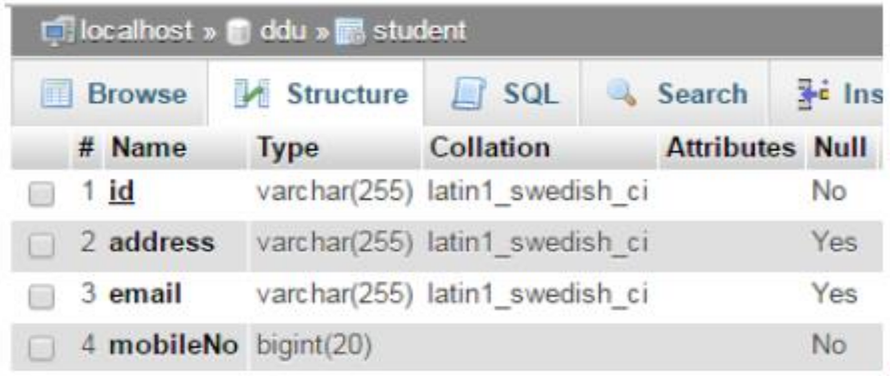


	#	Name	Type	Collation	Attributes	Null
☐	1	id	varchar(255)	latin1_swedish_ci		No
☐	2	address	varchar(255)	latin1_swedish_ci		Yes
☐	3	email	varchar(255)	latin1_swedish_ci		Yes
☐	4	mobileNo	bigint(20)			No
☐	5	FULL_NAME	varchar(255)	latin1_swedish_ci		No

Annotations for additional configurations

- @Transient
  - Hibernate will ignore that field while interacting with the database
- Structure of the table with following in Student_Info.java

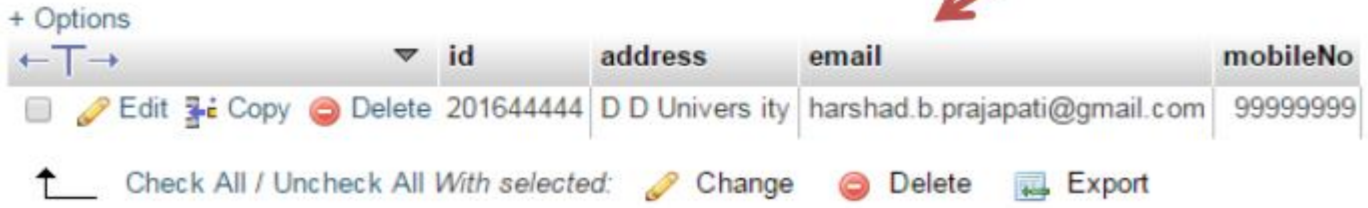
```
@Transient
@Column(name="FULL_NAME", nullable=false)
```



#	Name	Type	Collation	Attributes	Null
1	id	varchar(255)	latin1_swedish_ci		No
2	address	varchar(255)	latin1_swedish_ci		Yes
3	email	varchar(255)	latin1_swedish_ci		Yes
4	mobileNo	bigint(20)			No

The field FULL_NAME Is not present

- Records in the table



	id	address	email	mobileNo
Edit Copy Delete	201644444	D D Univers ity	harshad.b.prajapati@gmail.com	99999999

Annotations for additional configurations

- Add birthDate in Student_Info.java

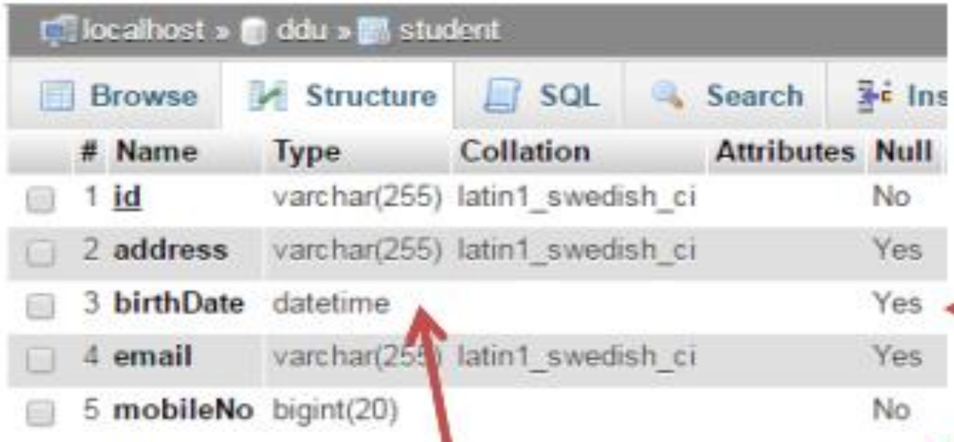
```
 private Date birthDate;
  public Date getBirthDate() {
      return birthDate;
   }
  public void setBirthDate(Date birthDate) {
      this.birthDate = birthDate;
   }
```

- In Main.java

```
student.setBirthDate(new Date());
```

Annotations for additional configurations

- Structure of the table

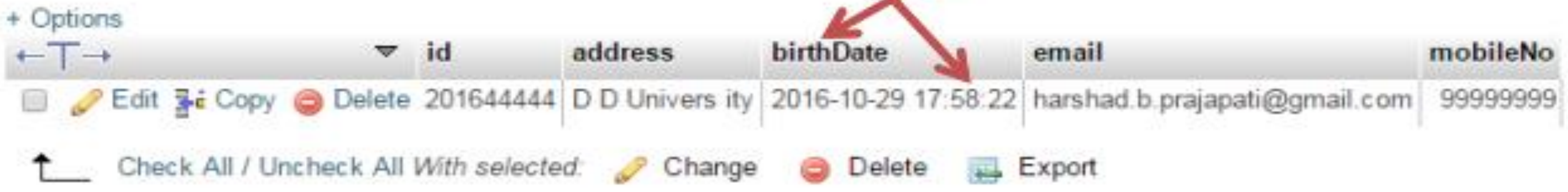


#	Name	Type	Collation	Attributes	Null
1	id	varchar(255)	latin1_swedish_ci		No
2	address	varchar(255)	latin1_swedish_ci		Yes
3	birthDate	datetime			Yes
4	email	varchar(255)	latin1_swedish_ci		Yes
5	mobileNo	bigint(20)			No

The birthDate field

It is with time stamp

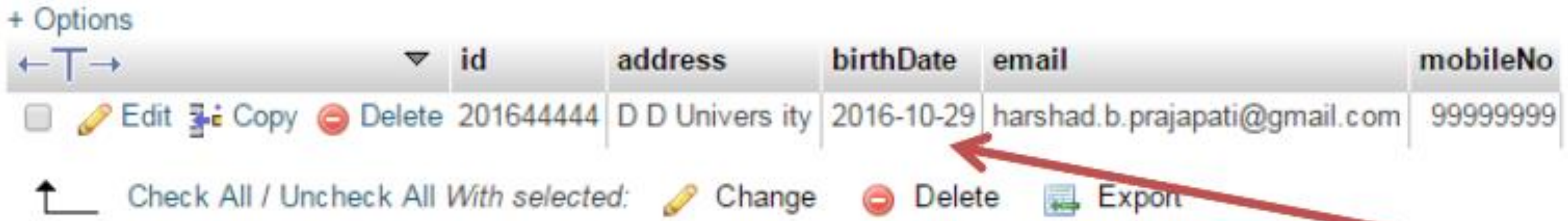
- Record in the table



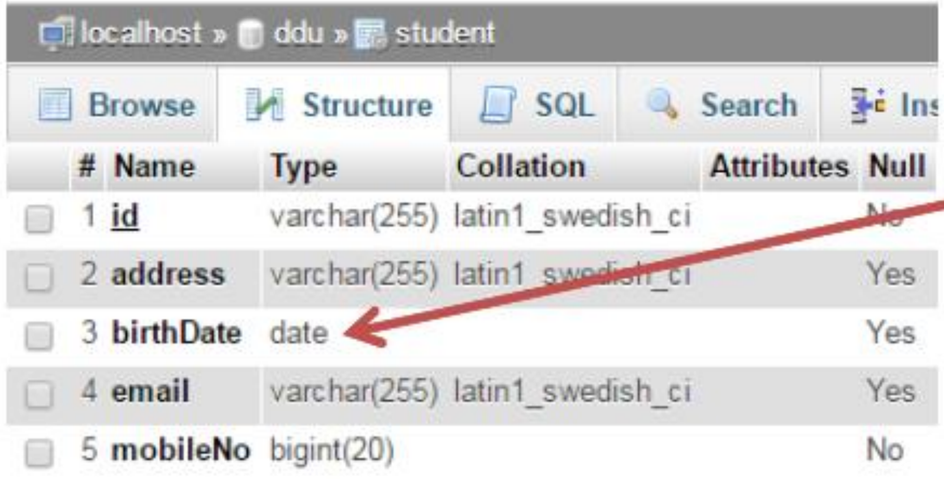
id	address	birthDate	email	mobileNo
201644444	D D Univers ity	2016-10-29 17:58:22	harshad.b.prajapati@gmail.com	99999999

- We want date without timestamp
  - Use @Temporal annotation
- In Student_Info.java class, do the following changes:

```
@Temporal(TemporalType.DATE)
private Date birthDate;
```



	id	address	birthDate	email	mobileNo
Edit Copy Delete	201644444	D D Univers ity	2016-10-29	harshad.b.prajapati@gmail.com	99999999



#	Name	Type	Collation	Attributes	Null
1	id	varchar(255)	latin1_swedish_ci		No
2	address	varchar(255)	latin1_swedish_ci		Yes
3	birthDate	date			Yes
4	email	varchar(255)	latin1_swedish_ci		Yes
5	mobileNo	bigint(20)			No

It is now without time stamp

Primary key auto generation

- We want to generate value of database field automatically for primary key (E.g., id)
- How hibernate generates value?
  - Using @GeneratedValue annotation

Primary key auto generation

- Do the following changes in the code in Student_Info.java

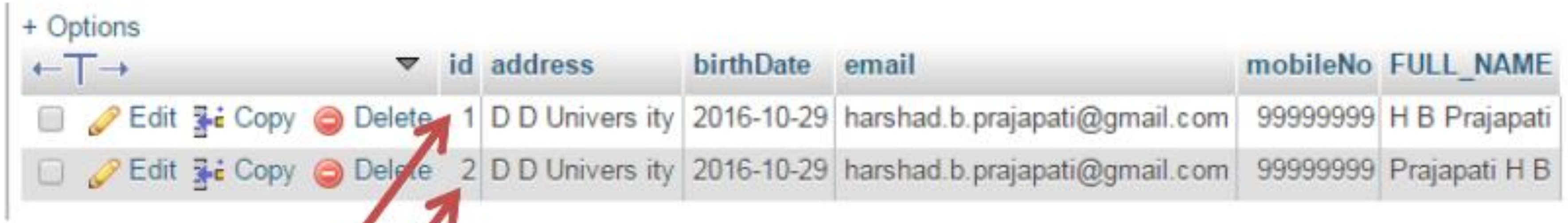
```
 @Id @GeneratedValue
   private int id;
 @Column(name="FULL_NAME", nullable=false)
   private String name;
```

- Do following changes in Main.java

```
Student_Info student1=new Student_Info();
student1.setName("H B Prajapati");
...
Student_Info student2=new Student_Info();
student2.setName("Prajapati H B");
...
 session.save(student1);
session.save(student2);
```

Primary key auto generation

- Two records with auto generated id value



+ Options

	id	address	birthDate	email	mobileNo	FULL_NAME
Edit Copy Delete	1	D D Univers ity	2016-10-29	harshad.b.prajapati@gmail.com	99999999	H B Prajapati
Edit Copy Delete	2	D D Univers ity	2016-10-29	harshad.b.prajapati@gmail.com	99999999	Prajapati H B

auto generated
id values for two
records

Primary key auto generation strategies

- Strategies for auto generation of values
  - AUTO
  - IDENTITY
  - SEQUENCE
  - TABLE
- Default is AUTO strategy
- AUTO
  - Hibernate would choose appropriate one (out of IDENTITY, SEQUENCE, and TABLE) for database,
  - e.g., for oracle, hibernate chooses SEQUENCE, as oracle does not support IDENTITY
  - e.g., for MySQL, hibernate chooses IDENTITY, as MySQL does not support SEQUENCE
- TABLE: some databases use separate TABLE object to create primary key