luv2code

Mapping Lists

List

An ordered collection of items

that allows duplicates

Use Case for List

Use Case for List

- Useful when
  - Order / index is important

Use Case for List

- Useful when
  - Order / index is important
  - Retrieve or manipulate items by order / index

Use Case for List

- Useful when
  - Order / index is important
  - Retrieve or manipulate items by order / index
- Examples

Use Case for List

- Useful when
  - Order / index is important
  - Retrieve or manipulate items by order / index
- Examples
  - List of emails in an inbox

Use Case for List

- Useful when
  - Order / index is important
  - Retrieve or manipulate items by order / index

- Examples
  - List of emails in an inbox

Use Case for List

- Useful when
  - Order / index is important
  - Retrieve or manipulate items by order / index

- Examples
  - List of emails in an inbox
  - Waiting list at a restaurant

Use Case for List

- Useful when
  - Order / index is important
  - Retrieve or manipulate items by order / index

- Examples
  - List of emails in an inbox
  - Waiting list at a restaurant

Student and Images

Student and Images

- A student will have a ***list*** of images (image file names)

Student and Images

- A student will have a ***list*** of images (image file names)
  - Keep track of order position

Student and Images

- A student will have a ***list*** of images (image file names)
  - Keep track of order position
  - Should allow duplicate images

Student and Images

- A student will have a ***list*** of images (image file names)
  - Keep track of order position
  - Should allow duplicate images

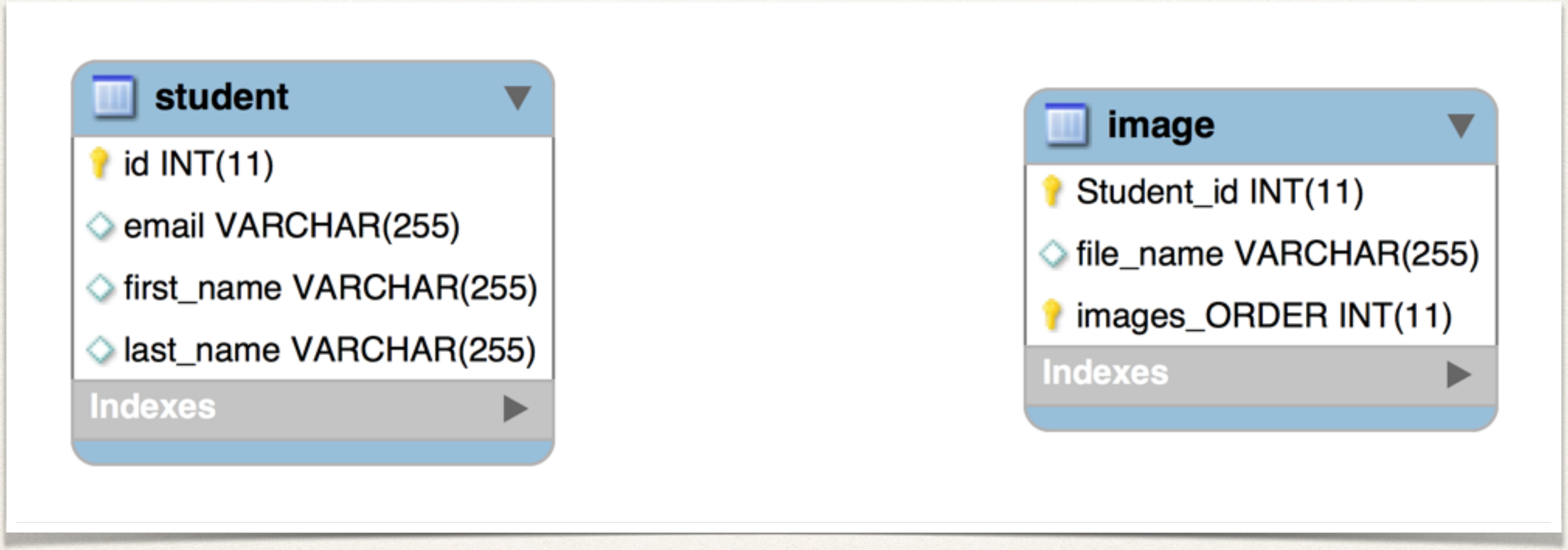
Student and Images

- A student will have a ***list*** of images (image file names)
  - Keep track of order position
  - Should allow duplicate images

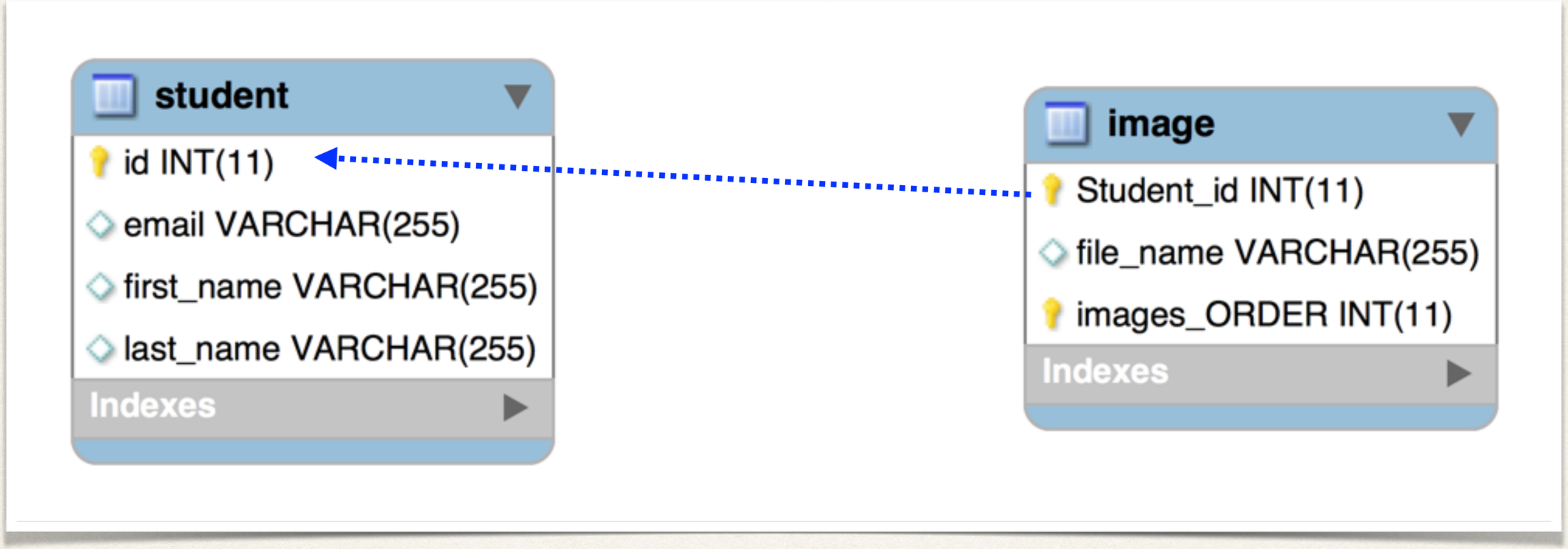
Student and Images

- A student will have a ***list*** of images (image file names)
  - Keep track of order position
  - Should allow duplicate images

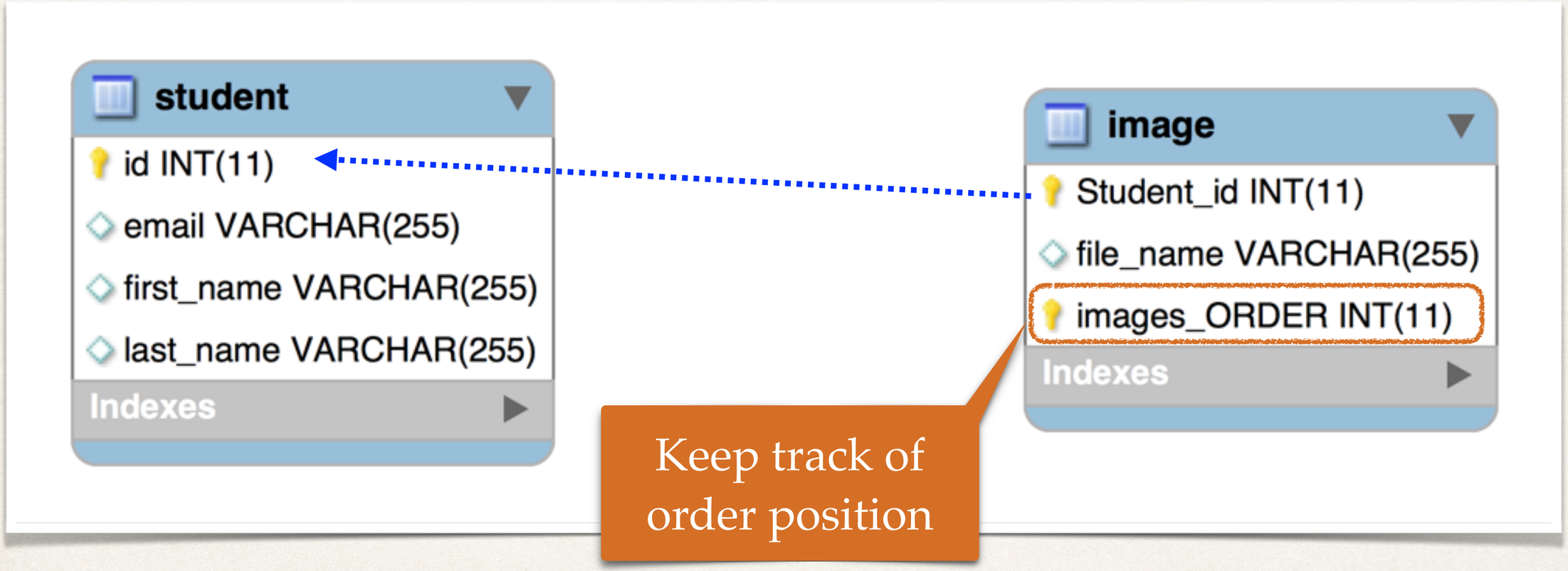
Database Diagram



Database Diagram



Database Diagram



Development Process

Step-By-Step

Development Process

Step-By-Step

1. Create database tables

Development Process

Step-By-Step

1. Create database tables

2. Map the element collection

Development Process

Step-By-Step

1. Create database tables

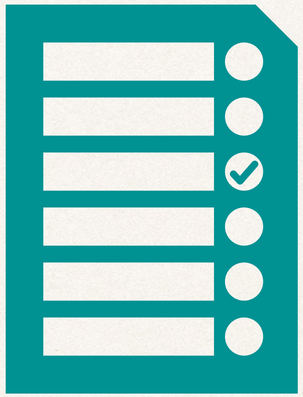
2. Map the element collection

3. Develop the main application

Step 1: Create database tables: `student`

Step 1: Create database tables: `student`

- Previously, we created database tables by running a SQL script



Step 1: Create database tables: `student`

Step 1: Create database tables: `student`

- Hibernate also provides an option to automagically create database tables

Step 1: Create database tables: `student`

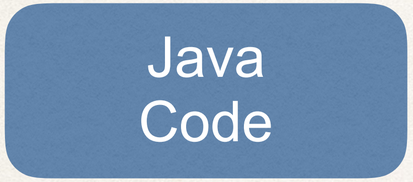
- Hibernate also provides an option to automagically create database tables
- Creates tables based on Java code with JPA/Hibernate annotations

Step 1: Create database tables: `student`

- Hibernate also provides an option to automagically create database tables
- Creates tables based on Java code with JPA/Hibernate annotations
- Useful for development and testing

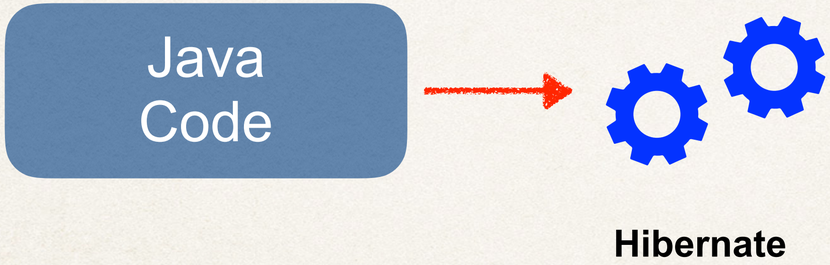
Step 1: Create database tables: `student`

- Hibernate also provides an option to automagically create database tables
- Creates tables based on Java code with JPA/Hibernate annotations
- Useful for development and testing



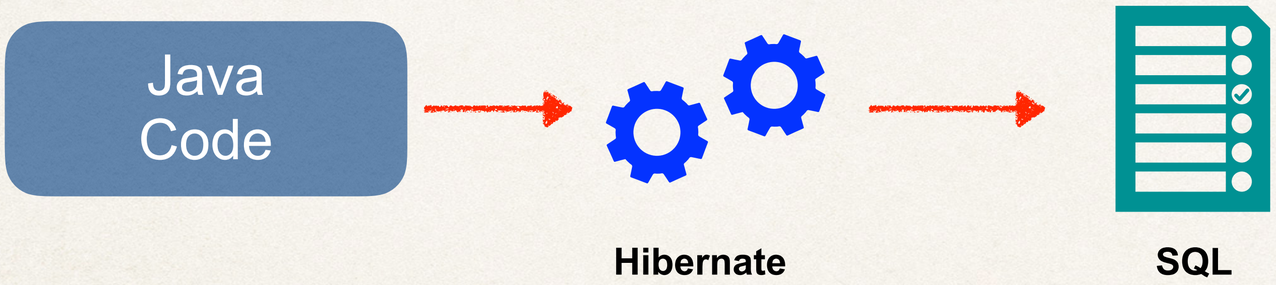
Step 1: Create database tables: `student`

- Hibernate also provides an option to automagically create database tables
- Creates tables based on Java code with JPA / Hibernate annotations
- Useful for development and testing



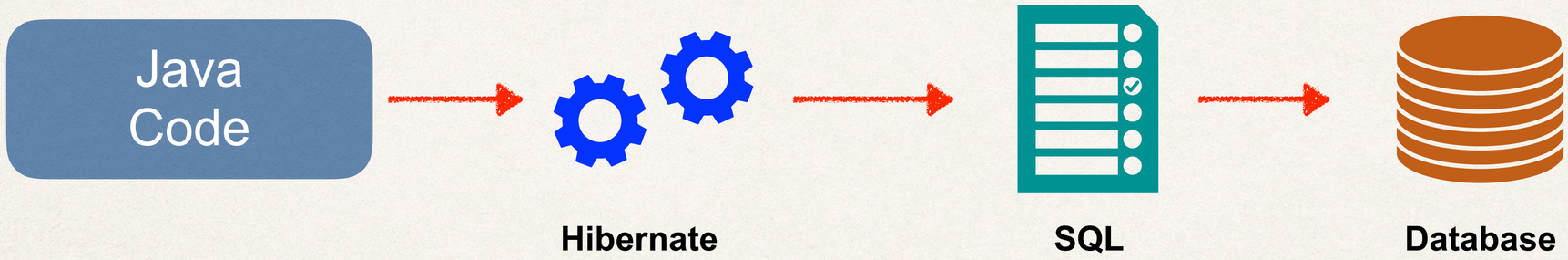
Step 1: Create database tables: `student`

- Hibernate also provides an option to automagically create database tables
- Creates tables based on Java code with JPA/Hibernate annotations
- Useful for development and testing



Step 1: Create database tables: `student`

- Hibernate also provides an option to automagically create database tables
- Creates tables based on Java code with JPA/Hibernate annotations
- Useful for development and testing



Hibernate Configuration

hbm: Hibernate mapping
ddl: Data definition language

Hibernate Configuration

- In Hibernate configuration file: **`hibernate.cfg.xml`**

hbm: Hibernate mapping
ddl: Data definition language

Hibernate Configuration

- In Hibernate configuration file: **`hibernate.cfg.xml`**

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

hbm: Hibernate mapping
ddl: Data definition language

Hibernate Configuration

- In Hibernate configuration file: **`hibernate.cfg.xml`**

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

- When you run your app, Hibernate will **drop** tables then **create** them

hbm: Hibernate mapping
ddl: Data definition language

Hibernate Configuration

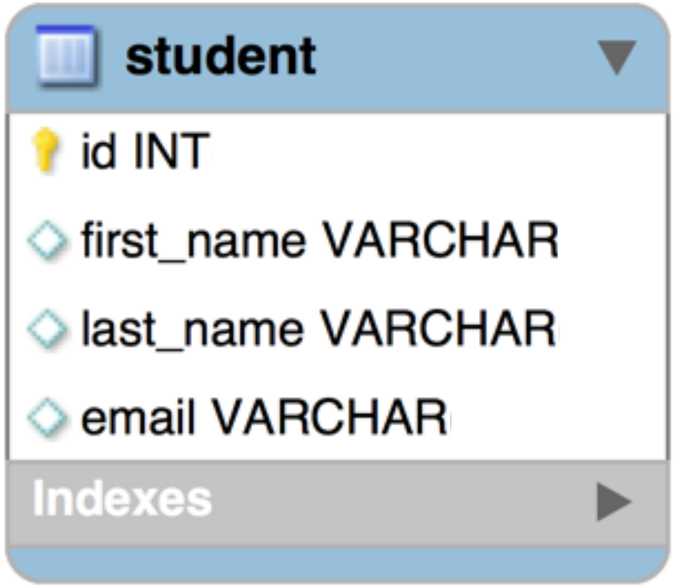
- In Hibernate configuration file: **hibernate.cfg.xml**

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

- When you run your app, Hibernate will **drop** tables then **create** them
- Based on the JPA/Hibernate annotations in your Java code

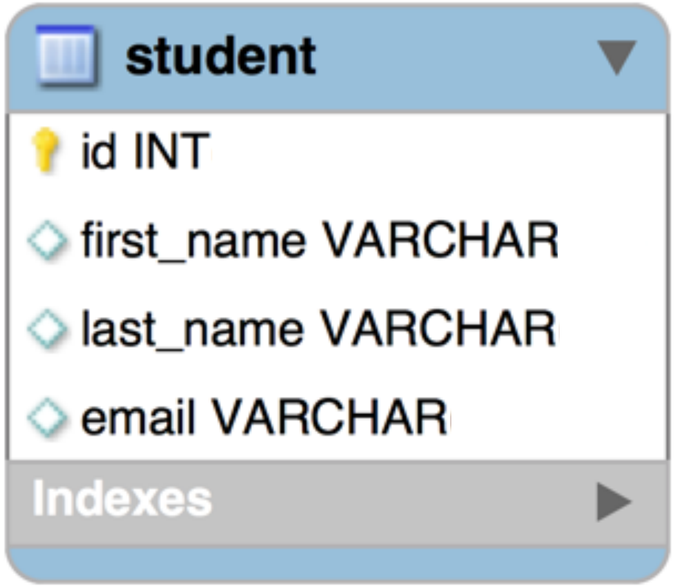
hbm: Hibernate mapping
ddl: Data definition language

Creating Tables based on Java Code



Creating Tables based on Java Code

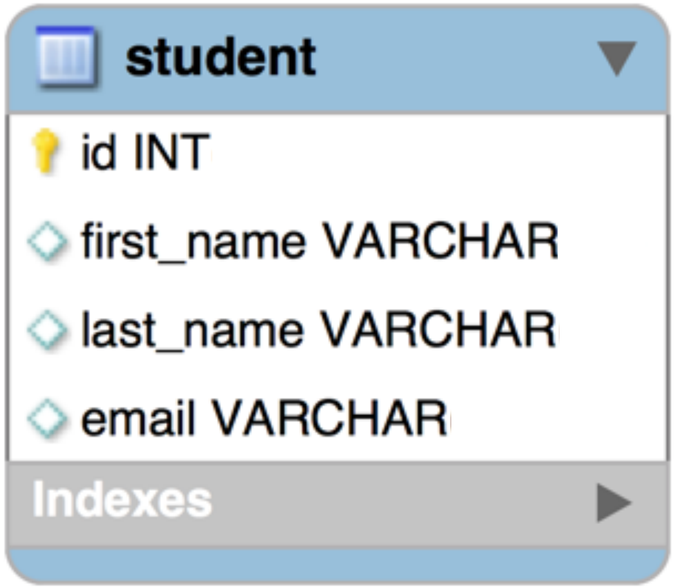
```
@Entity
@Table(name="student")
public class Student {

  @Id
  @GeneratedValue(strategy=GenerationType.IDENTITY)
  private int id;

  @Column(name="first_name")
  private String firstName;

  @Column(name="last_name")
  private String lastName;

  @Column(name="email")
  private String email;

  …
  // constructors, getters / setters
}
```



Creating Tables based on Java Code

```
@Entity
@Table(name="student")
public class Student {

  @Id
  @GeneratedValue(strategy=GenerationType.IDENTITY)
  private int id;

  @Column(name="first_name")
  private String firstName;

  @Column(name="last_name")
  private String lastName;

  @Column(name="email")
  private String email;

  …
  // constructors, getters / setters
}
```

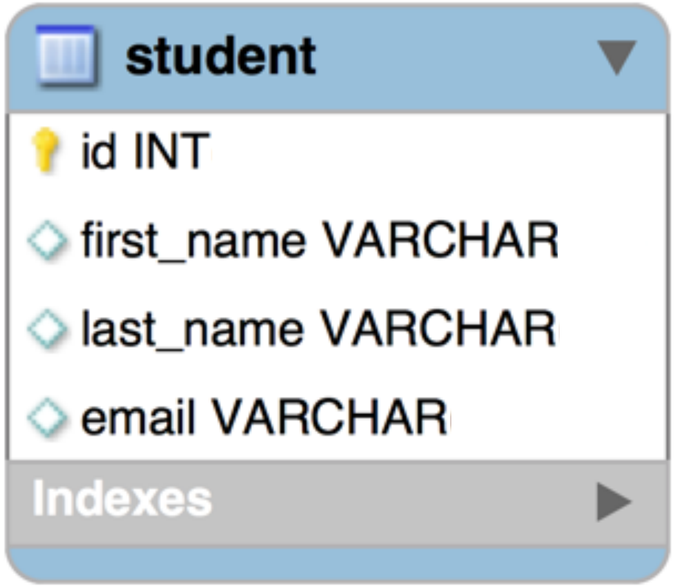
1



Creating Tables based on Java Code

```
@Entity
@Table(name="student")
public class Student {

  @Id
  @GeneratedValue(strategy=GenerationType.IDENTITY)
  private int id;

  @Column(name="first_name")
  private String firstName;

  @Column(name="last_name")
  private String lastName;

  @Column(name="email")
  private String email;

  …
  // constructors, getters / setters
}
```

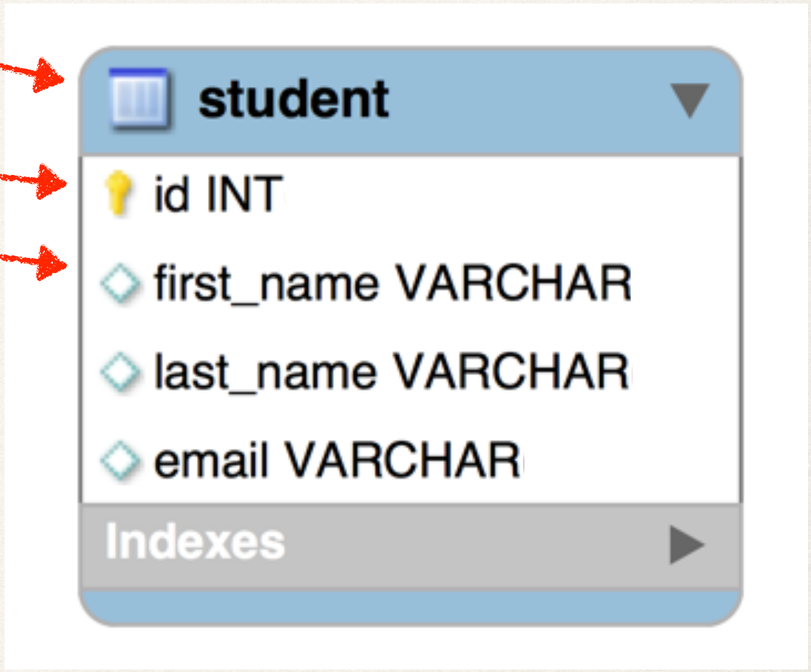
1



Creating Tables based on Java Code

```
@Entity
@Table(name="student")
public class Student {

  @Id
  @GeneratedValue(strategy=GenerationType.IDENTITY)
  private int id;

  @Column(name="first_name")
  private String firstName;

  @Column(name="last_name")
  private String lastName;

  @Column(name="email")
  private String email;

  …
  // constructors, getters / setters
}
```

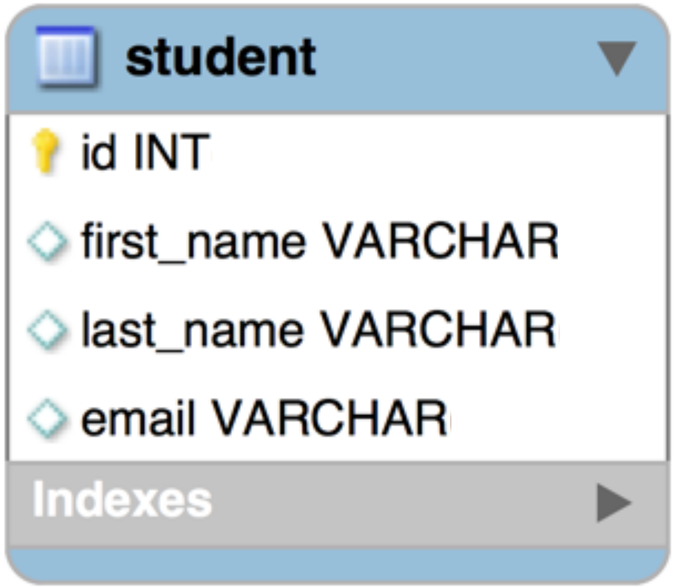
1



Creating Tables based on Java Code

```
@Entity
@Table(name="student")
public class Student {

  @Id
  @GeneratedValue(strategy=GenerationType.IDENTITY)
  private int id;

  @Column(name="first_name")
  private String firstName;

  @Column(name="last_name")
  private String lastName;

  @Column(name="email")
  private String email;

  …
  // constructors, getters / setters
}
```

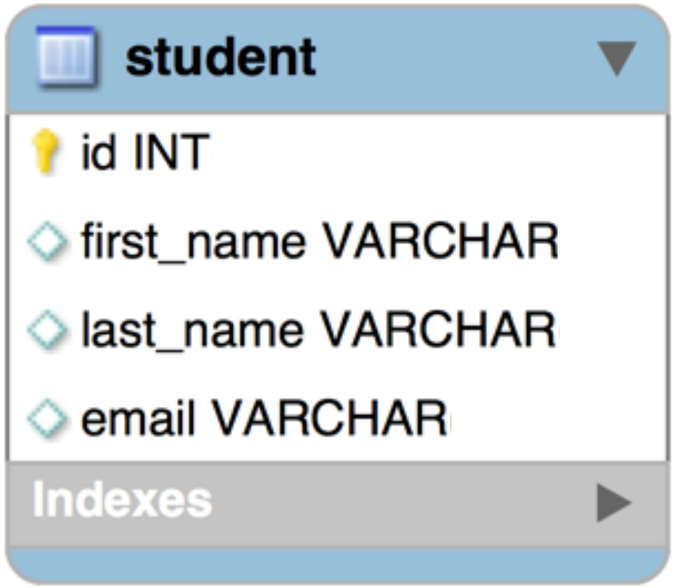
1



Creating Tables based on Java Code

```
@Entity
@Table(name="student")
public class Student {

   @Id
   @GeneratedValue(strategy=GenerationType.IDENTITY)
   private int id;

   @Column(name="first_name")
   private String firstName;

   @Column(name="last_name")
   private String lastName;

   @Column(name="email")
   private String email;


   …
   // constructors, getters / setters
}
```

1



Creating Tables based on Java Code

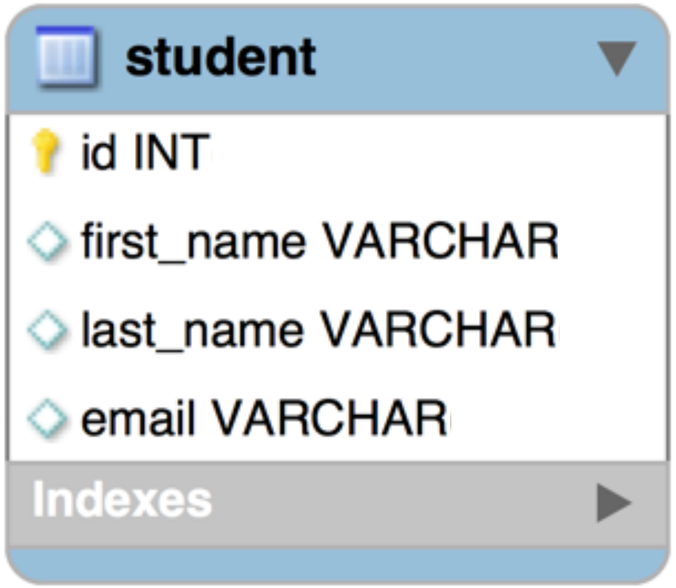
```java
@Entity
@Table(name="student")
public class Student {

  @Id
  @GeneratedValue(strategy=GenerationType.IDENTITY)
  private int id;

  @Column(name="first_name")
  private String firstName;

  @Column(name="last_name")
  private String lastName;

  @Column(name="email")
  private String email;

  …
  // constructors, getters / setters
}
```

1

2

Hibernate will generate and execute this

```
create table student (id integer not null auto_increment,
                      email varchar(255), first_name varchar(255),
                      last_name varchar(255), primary key (id))
```



Hibernate Configuration

Hibernate Configuration

```
<property name="hibernate.hbm2ddl.auto">PROPERTY-VALUE</property>
```

Property Value	Property Description

Hibernate Configuration

```
<property name="hibernate.hbm2ddl.auto">PROPERTY-VALUE</property>
```

Property Value	Property Description
none	No action will be performed

Hibernate Configuration

```
<property name="hibernate.hbm2ddl.auto">PROPERTY-VALUE</property>
```

Property Value	Property Description
`none`	No action will be performed
`create-only`	Database tables are only created

Hibernate Configuration

```
<property name="hibernate.hbm2ddl.auto">PROPERTY-VALUE</property>
```

Property Value	Property Description
`none`	No action will be performed
`create-only`	Database tables are only created
`drop`	Database tables are dropped

Hibernate Configuration

```
<property name="hibernate.hbm2ddl.auto">PROPERTY-VALUE</property>
```

Property Value	Property Description
none	No action will be performed
create-only	Database tables are only created
drop	Database tables are dropped

When database tables are dropped, all data is lost

Hibernate Configuration

```
<property name="hibernate.hbm2ddl.auto">PROPERTY-VALUE</property>
```

Property Value	Property Description
none	No action will be performed
create-only	Database tables are only created
drop	Database tables are dropped
create	Database tables are dropped followed by database tables creation

When database tables are dropped, all data is lost

Hibernate Configuration

```
<property name="hibernate.hbm2ddl.auto">PROPERTY-VALUE</property>
```

Property Value	Property Description
none	No action will be performed
create-only	Database tables are only created
drop	Database tables are dropped
create	Database tables are dropped followed by database tables creation
create-drop	Drop the database tables and recreate on SessionFactory startup. Additionally, drop the database tables on SessionFactory shutdown.

When database tables are dropped, all data is lost

Hibernate Configuration

```
<property name="hibernate.hbm2ddl.auto">PROPERTY-VALUE</property>
```

Property Value	Property Description
`none`	No action will be performed
`create-only`	Database tables are only created
`drop`	Database tables are dropped
`create`	Database tables are dropped followed by database tables creation
`create-drop`	Drop the database tables and recreate on SessionFactory startup. Additionally, drop the database tables on SessionFactory shutdown.
`validate`	Validate the database tables schema

Hibernate Configuration

```
<property name="hibernate.hbm2ddl.auto">PROPERTY-VALUE</property>
```

Property Value	Property Description
`none`	No action will be performed
`create-only`	Database tables are only created
`drop`	Database tables are dropped
`create`	Database tables are dropped followed by database tables creation
`create-drop`	Drop the database tables and recreate on SessionFactory startup. Additionally, drop the database tables on SessionFactory shutdown.
`validate`	Validate the database tables schema
`update`	Update the database tables schema

Ease of Use

Ease of Use

- For ease of development and testing, we'll use auto configuration

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

Ease of Use

- For ease of development and testing, we'll use auto configuration

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

- Database tables are **dropped** first and then **created** from scratch

Ease of Use

- For ease of development and testing, we'll use auto configuration

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

- Database tables are **dropped** first and then **created** from scratch

Note:

When database tables are dropped, all data is lost

Warning

Warning

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

Warning

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

- Don't do this on **Production** databases!!!

Warning

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

- Don't do this on **Production** databases!!!

- You normally don't want to **drop** your production data

Warning

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

- Don't do this on **Production** databases!!!

- You normally don't want to **drop** your production data
  - **All data is deleted!!!**

Warning

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

- Don't do this on **Production** databases!!!

- You normally don't want to **drop** your production data
  - **All data is deleted!!!**
- Instead for Production, you should have DBAs run SQL scripts

Step 2: Map the element collection

Step 2: Map the element collection

Step 2: Map the element collection


```
private List<String> images = new ArrayList<String>();
```

Annotation to Map Lists

Annotation to Map Lists

Annotation	Description

Annotation to Map Lists

Annotation	Description
`@OrderColumn`	**The name of the column to track element order / position.** **Name defaults to <property>_ORDER**

Annotation to Map Lists

Annotation	Description
`@OrderColumn`	**The name of the column to track element order / position.** **Name defaults to <property>_ORDER**

To override default name use:

```
@OrderColumn(name="my_data_position")
```

Annotation to Map Lists

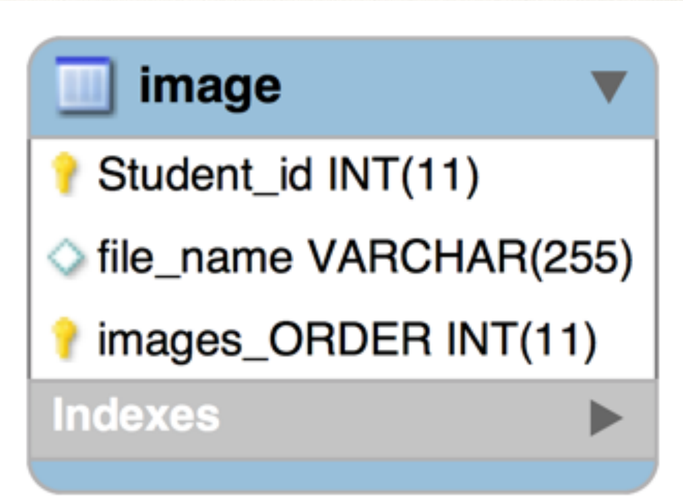
Annotation	Description
`@OrderColumn`	**The name of the column to track element order / position.** **Name defaults to <property>_ORDER**

To override default name use:

```
@OrderColumn(name="my_data_position")
```

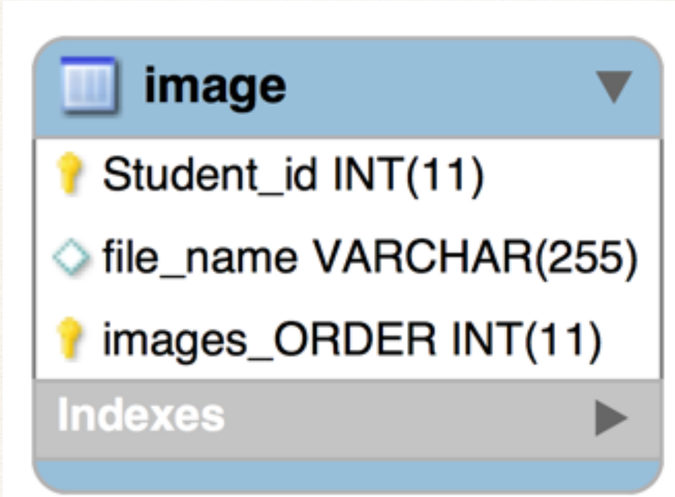
Any name

Mapping the Collection



Mapping the Collection

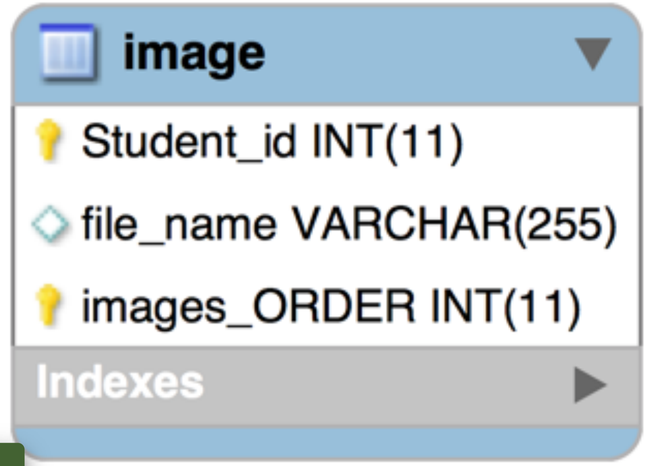
```
@Entity
@Table(name="student")
public class Student {









  private List<String> images = new ArrayList<String>();


  …
}
```



Mapping the Collection

```
@Entity
@Table(name="student")
public class Student {

  private List<String> images = new ArrayList<String>();

  ...
}
```

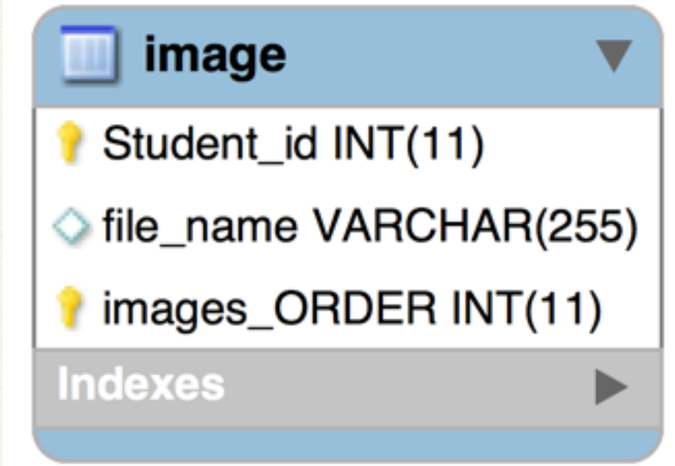
Use a **`List`** to track image file names



Mapping the Collection

```
@Entity
@Table(name="student")
public class Student {

  …

  @ElementCollection
  @CollectionTable(name="image")
  @OrderColumn
  @Column(name="file_name")
  private List<String> images = new ArrayList<String>();


  …
}
```

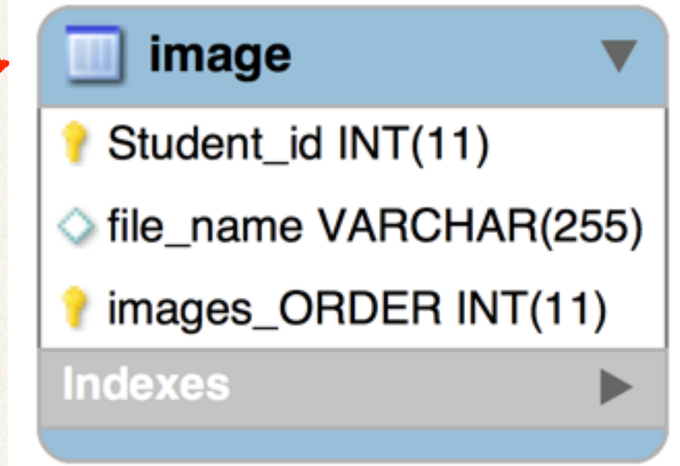
Tells Hibernate we are mapping a collection



Mapping the Collection

```
@Entity
@Table(name="student")
public class Student {

  …

  @ElementCollection
  @CollectionTable(name="image")
  @OrderColumn
  @Column(name="file_name")
  private List<String> images = new ArrayList<String>();


  …
}
```

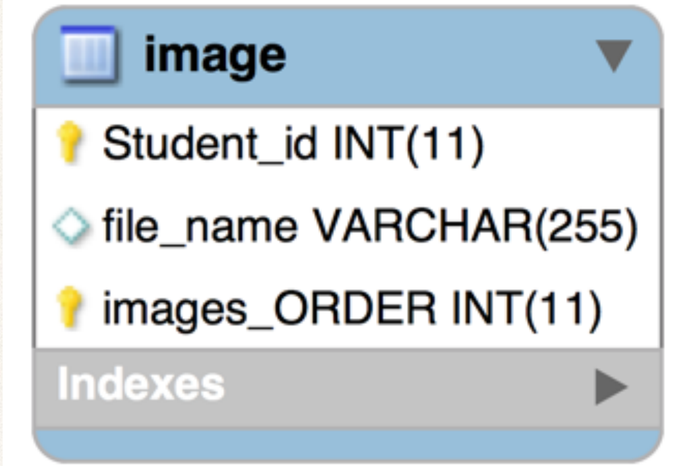
The name of the collection table



Mapping the Collection

```
@Entity
@Table
public

  …

  @ElementCollection
  @CollectionTable(name="image")
  @OrderColumn
  @Column(name="file_name")
  private List<String> images = new ArrayList<String>();


  …
}
```

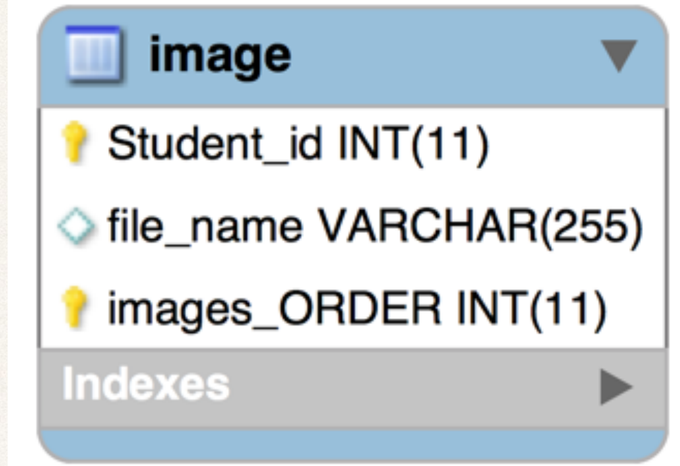
Column to track element order
Name defaults to: <property>_ORDER



Mapping the Collection

```
@Entity
@Table
public

  …

  @ElementCollection
  @CollectionTable(name="image")
  @OrderColumn
  @Column(name="file_name")
  private List<String> images = new ArrayList<String>();



  …
}
```

Column to track element order
Name defaults to: <property>_ORDER



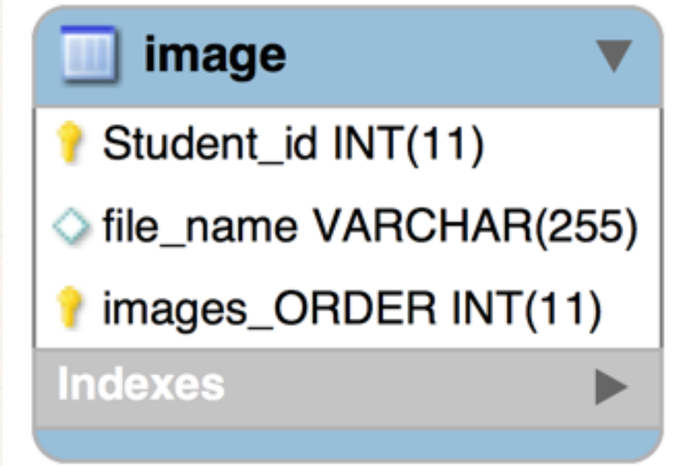
In this example
<property> is images

Mapping the Collection

To override default use:
`@OrderColumn(name="my_data_position")`

```
@Entity
@Table
public

  …

  @ElementCollection
  @CollectionTable(name="image")
  @OrderColumn
  @Column(name="file_name")
  private List<String> images = new ArrayList<String>();



  …
}
```

Column to track element order
Name defaults to: `<property>_ORDER`

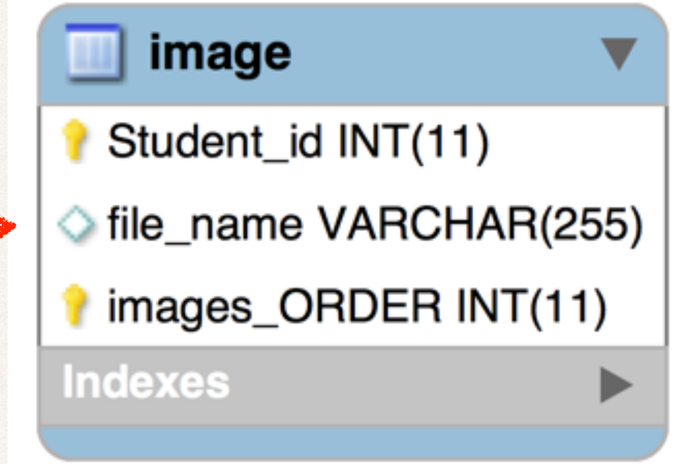


In this example
`<property>` is `images`

Mapping the Collection

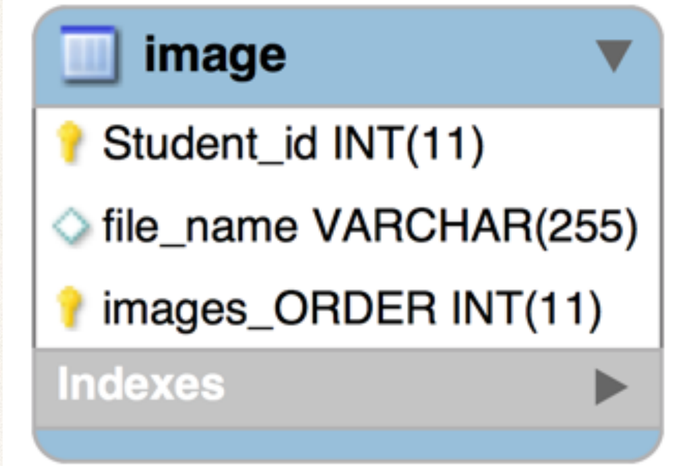
```
@Entity
@Table(name="student")
public class Student {

  …

  @ElementCollection
  @CollectionTable(name="image")
  @OrderColumn
  @Column(name="file_name")
  private List<String> images = new ArrayList<String>();


  …
}
```

The column for the image file name



Mapping the Collection

```
@Entity
@Table(name="student")
public class Student {

  …

  @ElementCollection
  @CollectionTable(name="image")
  @OrderColumn
  @Column(name="file_name")
  private List<String> images = new ArrayList<String>();


  …
}
```



Step 3: Develop the main application

Step 3: Develop the main application

```
// create the object
Student tempStudent =  new Student("John", "Doe", "john@luv2code.com");
List<String> theImages = tempStudent.getImages();
```

Get a reference
to the images **`List`**

Step 3: Develop the main application

```
// create the object
Student tempStudent =  new Student("John", "Doe", "john@luv2code.com");
List<String> theImages = tempStudent.getImages();

theImages.add("photo1.jpg");
theImages.add("photo2.jpg");
theImages.add("photo3.jpg");
theImages.add("photo4.jpg");
theImages.add("photo4.jpg");
```

Now, let's add to the images **`List`**

Step 3: Develop the main application

```
// create the object
Student tempStudent =  new Student("John", "Doe", "john@luv2code.com");
List<String> theImages = tempStudent.getImages();

theImages.add("photo1.jpg");
theImages.add("photo2.jpg");
theImages.add("photo3.jpg");
theImages.add("photo4.jpg");
theImages.add("photo4.jpg");
```

Now, let's add to the images **`List`**

Remember, duplicates are allowed in **`List`**

Step 3: Develop the main application

```
// create the object
Student tempStudent =  new Student("John", "Doe", "john@luv2code.com");
List<String> theImages = tempStudent.getImages();

theImages.add("photo1.jpg");
theImages.add("photo2.jpg");
theImages.add("photo3.jpg");
theImages.add("photo4.jpg");
theImages.add("photo4.jpg");
```

Now, let's add to the images **`List`**

Step 3: Develop the main application

```
// create the object
Student tempStudent =  new Student("John", "Doe", "john@luv2code.com");
List<String> theImages = tempStudent.getImages();

theImages.add("photo1.jpg");
theImages.add("photo2.jpg");
theImages.add("photo3.jpg");
theImages.add("photo4.jpg");
theImages.add("photo4.jpg");

// start a transaction
session.beginTransaction();
```

Step 3: Develop the main application

```
// create the object
Student tempStudent =  new Student("John", "Doe", "john@luv2code.com");
List<String> theImages = tempStudent.getImages();

theImages.add("photo1.jpg");
theImages.add("photo2.jpg");
theImages.add("photo3.jpg");
theImages.add("photo4.jpg");
theImages.add("photo4.jpg");

// start a transaction
session.beginTransaction();

// save the object
System.out.println("Saving the student and images..");
session.persist(tempStudent);
```

Step 3: Develop the main application

```
// create the object
Student tempStudent =  new Student("John", "Doe", "john@luv2code.com");
List<String> theImages = tempStudent.getImages();

theImages.add("photo1.jpg");
theImages.add("photo2.jpg");
theImages.add("photo3.jpg");
theImages.add("photo4.jpg");
theImages.add("photo4.jpg");

// start a transaction
session.beginTransaction();

// save the object
System.out.println("Saving the student and images..");
session.persist(tempStudent);

// commit the transaction
session.getTransaction().commit();
```

Run the App

Run the App

Console output

```
Hibernate: insert into student (email, first_name, last_name) values (?, ?, ?)
Hibernate: insert into image (Student_id, images_ORDER, file_name) values (?, ?, ?)
Hibernate: insert into image (Student_id, images_ORDER, file_name) values (?, ?, ?)
Hibernate: insert into image (Student_id, images_ORDER, file_name) values (?, ?, ?)
Hibernate: insert into image (Student_id, images_ORDER, file_name) values (?, ?, ?)
Hibernate: insert into image (Student_id, images_ORDER, file_name) values (?, ?, ?)
```

Run the App

Console output

```
Hibernate: insert into student (email, first_name, last_name) values (?, ?, ?)
Hibernate: insert into image (Student_id, images_ORDER, file_name) values (?, ?, ?)
Hibernate: insert into image (Student_id, images_ORDER, file_name) values (?, ?, ?)
Hibernate: insert into image (Student_id, images_ORDER, file_name) values (?, ?, ?)
Hibernate: insert into image (Student_id, images_ORDER, file_name) values (?, ?, ?)
Hibernate: insert into image (Student_id, images_ORDER, file_name) values (?, ?, ?)
```

Table: student

id	email	first_name	last_name
1	john@luv2code.com	John	Doe

Run the App

Console output

```
Hibernate: insert into student (email, first_name, last_name) values (?, ?, ?)
Hibernate: insert into image (Student_id, images_ORDER, file_name) values (?, ?, ?)
Hibernate: insert into image (Student_id, images_ORDER, file_name) values (?, ?, ?)
Hibernate: insert into image (Student_id, images_ORDER, file_name) values (?, ?, ?)
Hibernate: insert into image (Student_id, images_ORDER, file_name) values (?, ?, ?)
Hibernate: insert into image (Student_id, images_ORDER, file_name) values (?, ?, ?)
```

Table: student

id	email	first_name	last_name
1	john@luv2code.com	John	Doe

Table: image

Student_id	file_name	images_ORDER
1	photo1.jpg	0
1	photo2.jpg	1
1	photo3.jpg	2
1	photo4.jpg	3
1	photo4.jpg	4

Run the App

Console output

```
Hibernate: insert into student (email, first_name, last_name) values (?, ?, ?)
Hibernate: insert into image (Student_id, images_ORDER, file_name) values (?, ?, ?)
Hibernate: insert into image (Student_id, images_ORDER, file_name) values (?, ?, ?)
Hibernate: insert into image (Student_id, images_ORDER, file_name) values (?, ?, ?)
Hibernate: insert into image (Student_id, images_ORDER, file_name) values (?, ?, ?)
Hibernate: insert into image (Student_id, images_ORDER, file_name) values (?, ?, ?)
```

Table: student

id	email	first_name	last_name
1	john@luv2code.com	John	Doe

Table: image

Student_id	file_name	images_ORDER
1	photo1.jpg	0
1	photo2.jpg	1
1	photo3.jpg	2
1	photo4.jpg	3
1	photo4.jpg	4

Run the App

Console output

```
Hibernate: insert into student (email, first_name, last_name) values (?, ?, ?)
Hibernate: insert into image (Student_id, images_ORDER, file_name) values (?, ?, ?)
Hibernate: insert into image (Student_id, images_ORDER, file_name) values (?, ?, ?)
Hibernate: insert into image (Student_id, images_ORDER, file_name) values (?, ?, ?)
Hibernate: insert into image (Student_id, images_ORDER, file_name) values (?, ?, ?)
Hibernate: insert into image (Student_id, images_ORDER, file_name) values (?, ?, ?)
```

Table: student

id	email	first_name	last_name
1	john@luv2code.com	John	Doe

Column to track element order

Table: image

Student_id	file_name	images_ORDER
1	photo1.jpg	0
1	photo2.jpg	1
1	photo3.jpg	2
1	photo4.jpg	3
1	photo4.jpg	4