luv2code

Mapping - Maps

Map

An object that maps unique keys to values

Key is used to store and retrieve values

Can not contain duplicate keys

Use Case for Maps

Use Case for Maps

- Useful when you want to access data via a key rather than integer index

Use Case for Maps

- Useful when you want to access data via a key rather than integer index

- Examples

Use Case for Maps

- Useful when you want to access data via a key rather than integer index
- Examples
  - Map of error codes and error messages

Key	Value
20	Disk Full
30	Syntax Error
40	Account Blocked

Use Case for Maps

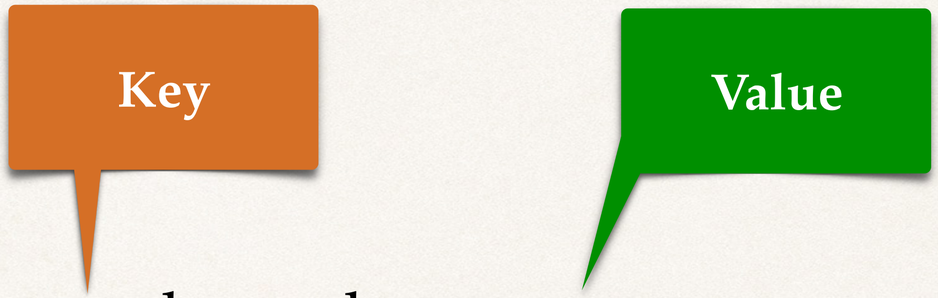
- Useful when you want to access data via a key rather than integer index

- Examples
  - Map of error codes and error messages
  - Map of country codes and country names

Key	Value
IN	India
DE	Germany
CR	Costa Rica

Use Case for Maps

- Useful when you want to access data via a key rather than integer index



- Examples
  - Map of error codes and error messages
  - Map of country codes and country names
  - Map of image file names and image descriptions

Key	Value
photo1.jpg	Photo 1
photo2.jpg	Photo 2
photo3.jpg	Photo 3

Student and Images

Student and Images

- A student will have a ***map*** of images

Student and Images

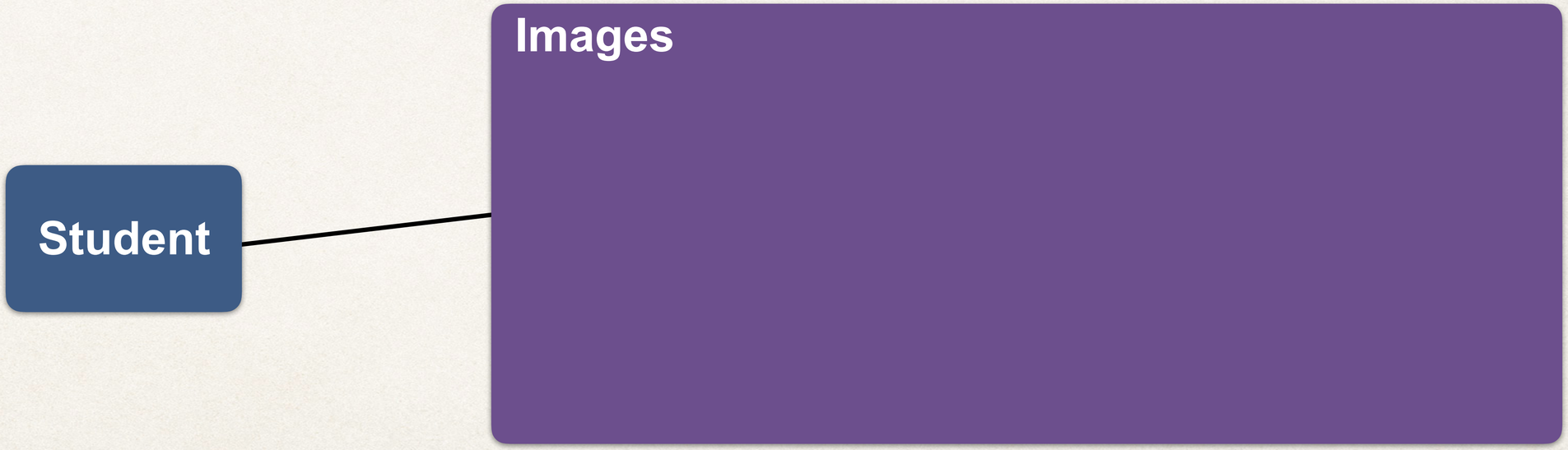
- A student will have a ***map*** of images
  - Each image will have a ***file name*** and a ***description***

Student and Images

- A student will have a ***map*** of images
  - Each image will have a ***file name*** and a ***description***
  - Duplicate file names are not allowed

Student and Images

- A student will have a ***map*** of images
  - Each image will have a ***file name*** and a ***description***
  - Duplicate file names are not allowed

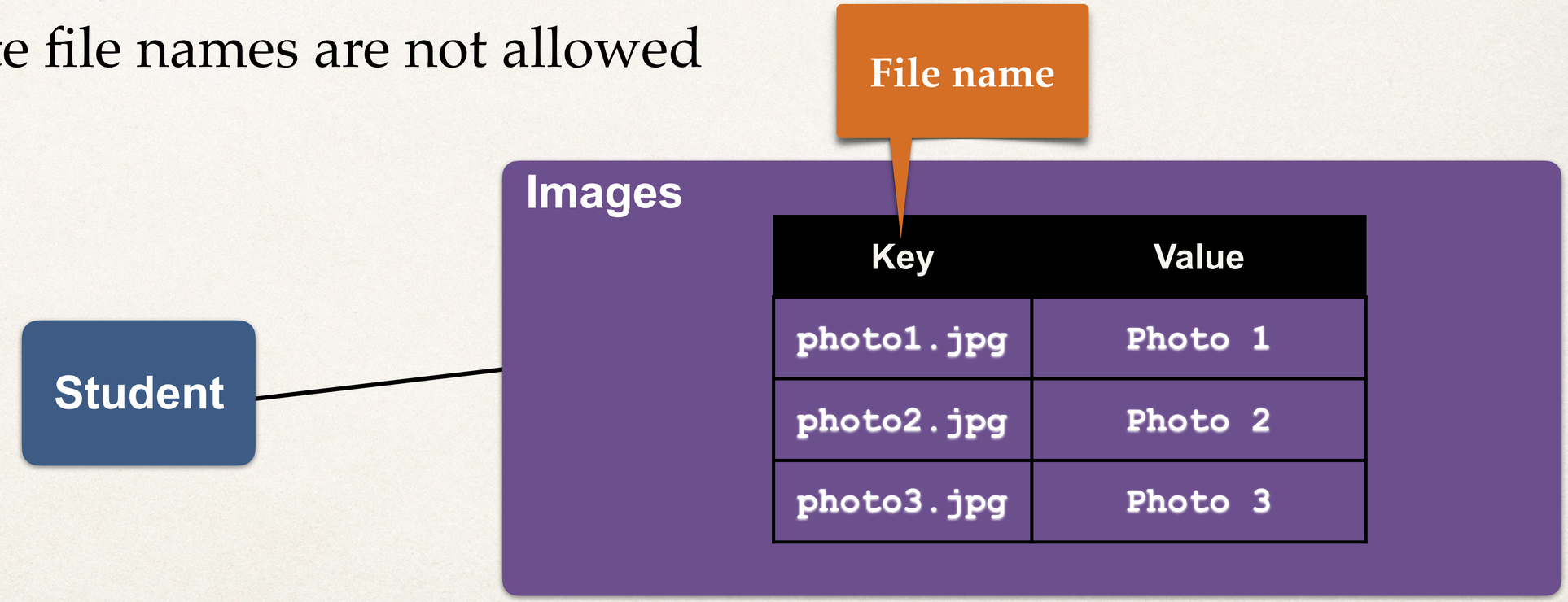


Student and Images

- A student will have a ***map*** of images
  - Each image will have a ***file name*** and a ***description***
  - Duplicate file names are not allowed

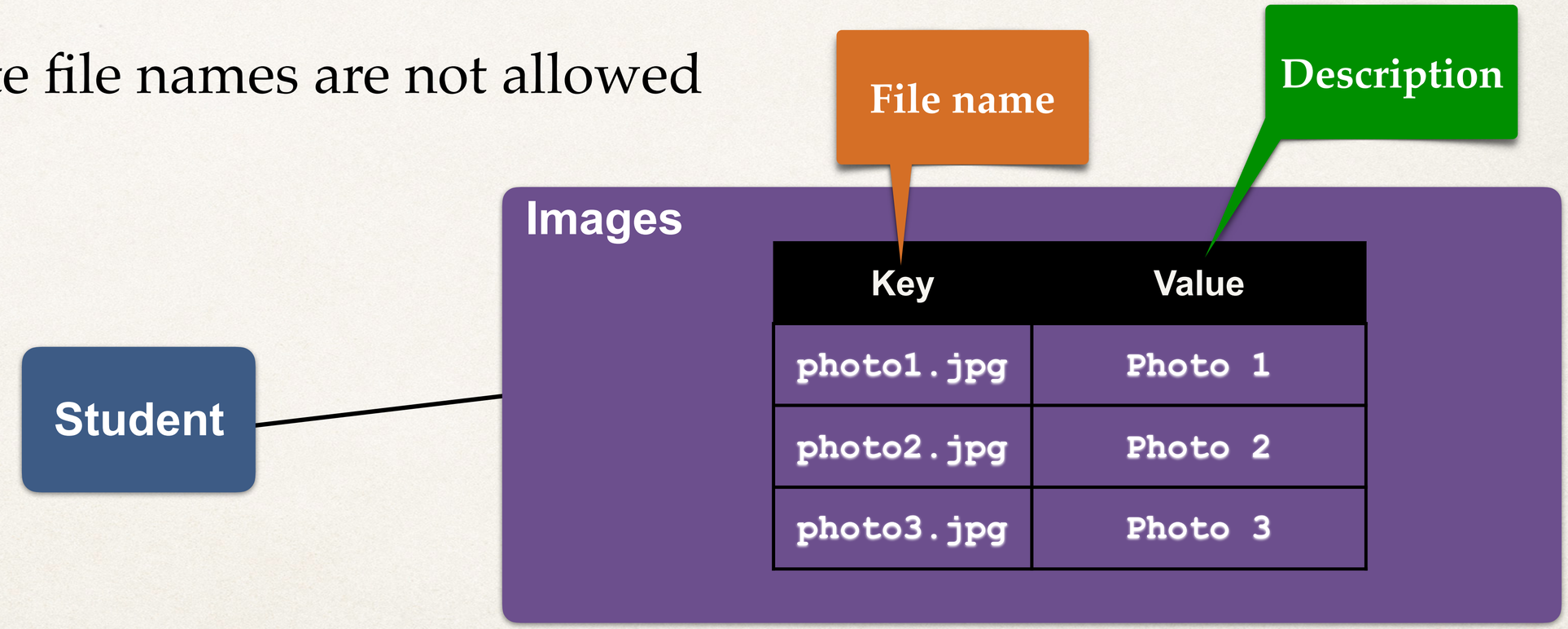
Student and Images

- A student will have a ***map*** of images
  - Each image will have a ***file name*** and a ***description***
  - Duplicate file names are not allowed

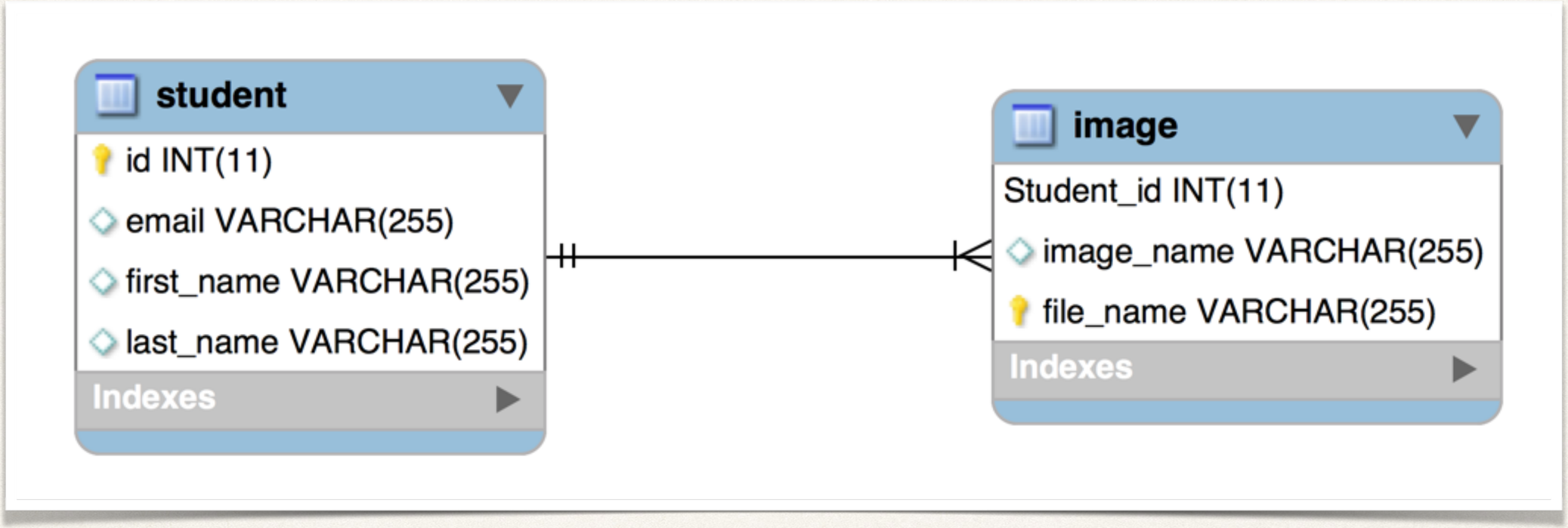


Student and Images

- A student will have a ***map*** of images
  - Each image will have a ***file name*** and a ***description***
  - Duplicate file names are not allowed



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. Annotate the Map

Development Process

Step-By-Step

1. Create database tables

2. Annotate the Map

3. Develop the main application

Step 1: Create database tables

Step 1: Create database tables

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

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

Step 1: Create database tables

- 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

Step 2: Annotate the Map

Step 2: Annotate the Map

Step 2: Annotate the Map


```
private Map<String, String> images = new HashMap<String, String>();
```

Annotation for Maps

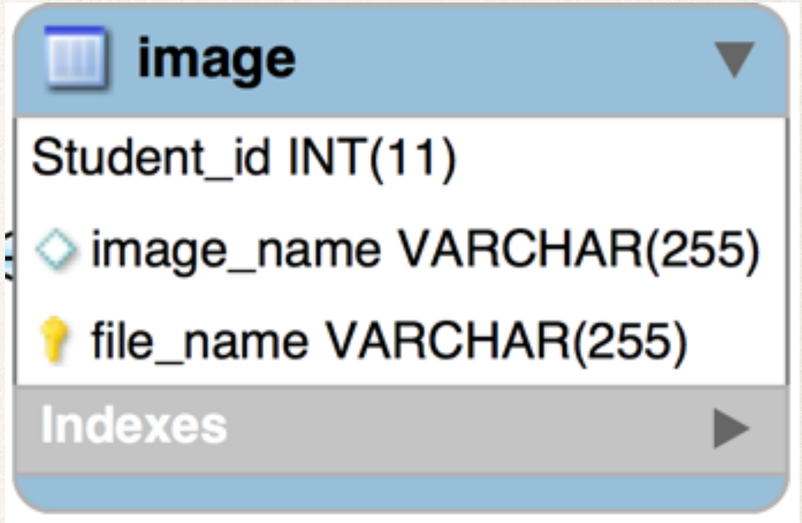
Annotation for Maps

Annotation	Description

Annotation for Maps

Annotation	Description
`@MapKeyColumn`	**The name of the key column for a map** **Name defaults to <property>_KEY**

Step 2: Annotate the Map



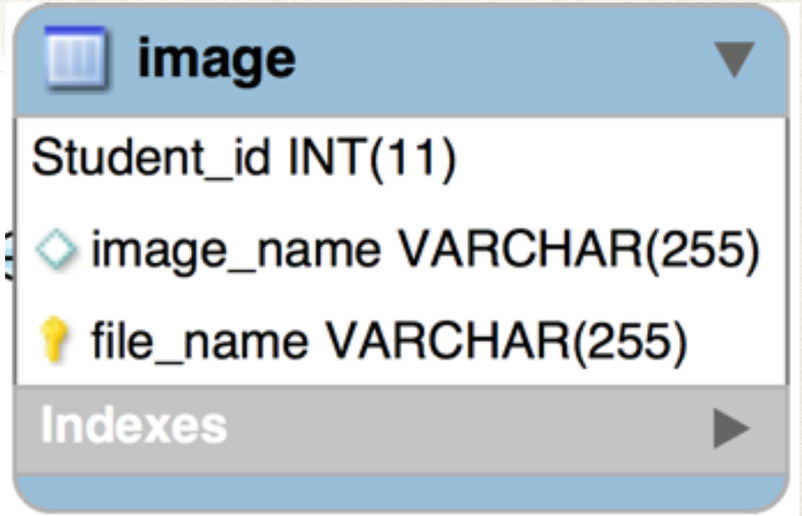
Step 2: Annotate the Map

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

  private Map<String, String> images = new HashMap<String, String>();

  …
}
```

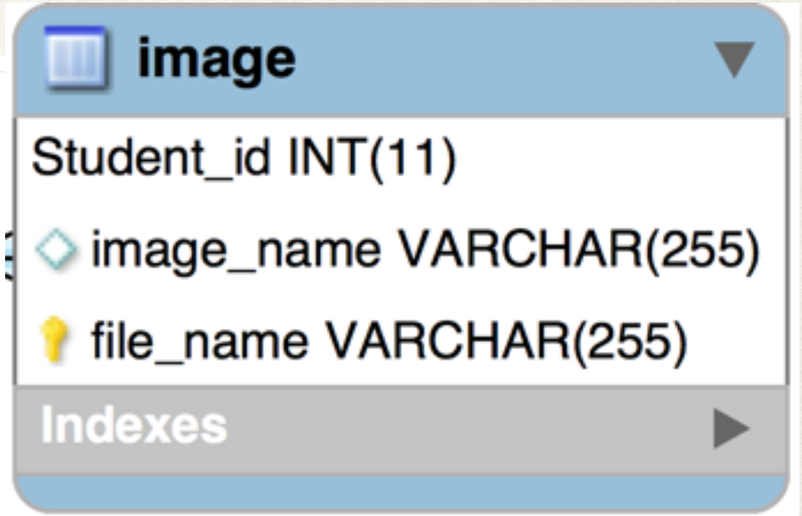


Step 2: Annotate the Map

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

  private Map<String, String> images = new HashMap<String, String>();

  …
}
```



Use a **Map** to track
image file names and descriptions

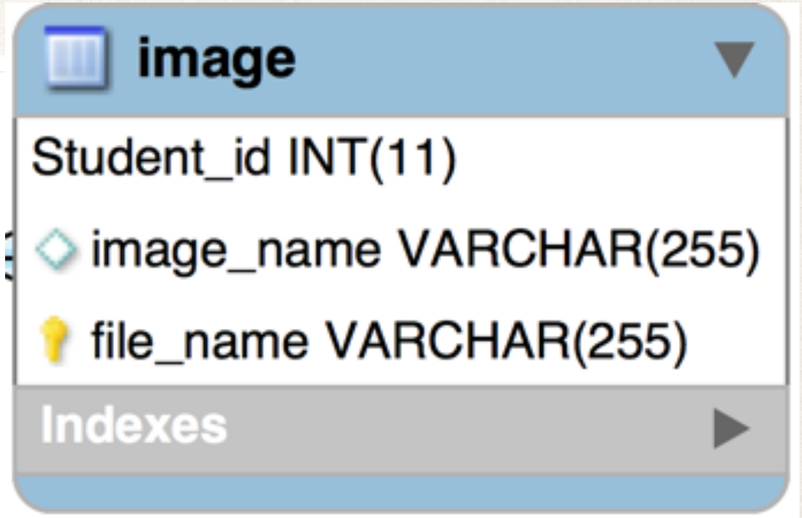
Step 2: Annotate the Map

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

  …

  @ElementCollection
  @CollectionTable(name="image")
  @MapKeyColumn(name="file_name")
  @Column(name="image_name")
  private Map<String, String> images = new HashMap<String, String>();


  …
}
```



Step 2: Annotate the Map

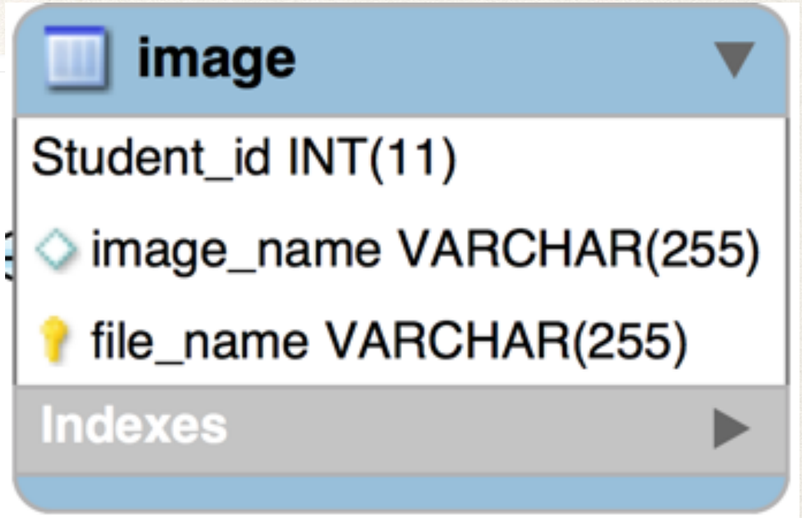
```
@Entity
@Table(name="student")
public class Student {

  …

  @ElementCollection
  @CollectionTable(name="image")
  @MapKeyColumn(name="file_name")
  @Column(name="image_name")
  private Map<String, String> images = new HashMap<String, String>();


  …
}
```

Column for map key



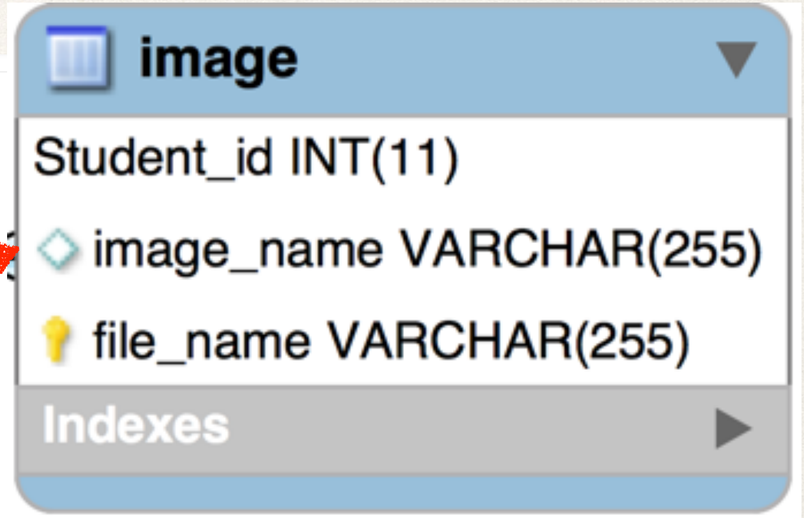
Step 2: Annotate the Map

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

  …

  @ElementCollection
  @CollectionTable(name="image")
  @MapKeyColumn(name="file_name")
  @Column(name="image_name")
  private Map<String, String> images = new HashMap<String, String>();


  …
}
```



The column for map value

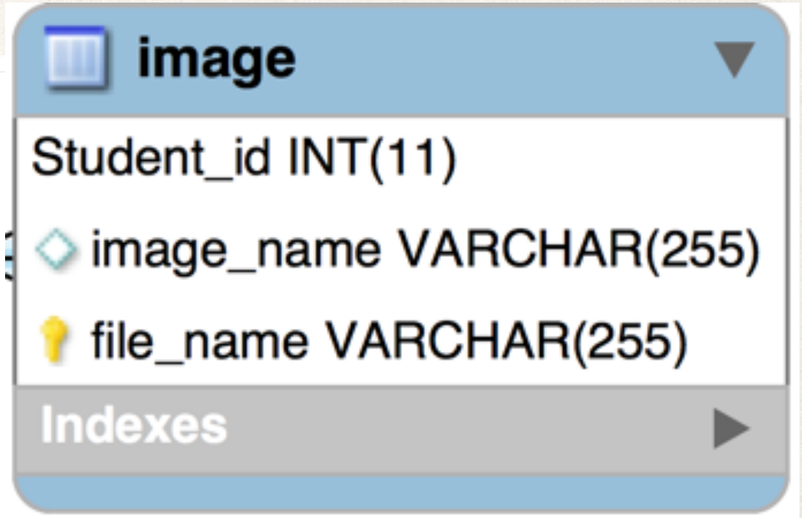
Step 2: Annotate the Map

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

  …

  @ElementCollection
  @CollectionTable(name="image")
  @MapKeyColumn(name="file_name")
  @Column(name="image_name")
  private Map<String, String> images = new HashMap<String, 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");
Map<String, String> theImages = tempStudent.getImages();
```

Get a reference to the images **Map**

Step 3: Develop the main application

```
// create the ob
Student tempStud          Stude          Doe", "john@luv2code.com");
Map<String, String> theImages = tempStudent.getImages();

theImages.put("photo1.jpg", "Photo 1");
theImages.put("photo2.jpg", "Photo 2");
theImages.put("photo3.jpg", "Photo 3");
```

Key

Value

Now, let's add to the images **Map**

Step 3: Develop the main application

```
// create the ob
Student tempStud      = new Stude          Doe", "john@luv2code.com");
Map<String, String> theImages = tempStudent.getImages();

theImages.put("photo1.jpg", "Photo 1");
theImages.put("photo2.jpg", "Photo 2");
theImages.put("photo3.jpg", "Photo 3");
```

Key

Value

File name

Description

Images

Key	Value
photo1.jpg	Photo 1
photo2.jpg	Photo 2
photo3.jpg	Photo 3

Step 3: Develop the main application

Key

Value

```
// create the ob
Student tempStu      = new Stude          Doe", "john@luv2code.com");
Map<String, String> theImages = tempStudent.getImages();

theImages.put("photo1.jpg", "Photo 1");
theImages.put("photo2.jpg", "Photo 2");
theImages.put("photo3.jpg", "Photo 3");


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

Step 3: Develop the main application

Key

Value

```
// create the ob
Student tempStud          Stude          Doe", "john@luv2code.com");
Map<String, String> theImages = tempStudent.getImages();

theImages.put("photo1.jpg", "Photo 1");
theImages.put("photo2.jpg", "Photo 2");
theImages.put("photo3.jpg", "Photo 3");


// 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

Key

Value

```
// create the ob
Student tempStu      Stude      Doe", "john@luv2code.com");
Map<String, String> theImages = tempStudent.getImages();

theImages.put("photo1.jpg", "Photo 1");
theImages.put("photo2.jpg", "Photo 2");
theImages.put("photo3.jpg", "Photo 3");


// 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, file_name, image_name) values (?, ?, ?)
Hibernate: insert into image (Student_id, file_name, image_name) values (?, ?, ?)
Hibernate: insert into image (Student_id, file_name, image_name) values (?, ?, ?)
```

Run the App

Console output

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

Table: student

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

Table: image

Student_id	file_name	image_name
1	photo1.jpg	Photo 1
1	photo2.jpg	Photo 2
1	photo3.jpg	Photo 3

```
select Student_id, file_name, image_name from image
```

Run the App

Console output

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

Table: student

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

Key

Value

Table: image

Student_id	file_name	image_name
1	photo1.jpg	Photo 1
1	photo2.jpg	Photo 2
1	photo3.jpg	Photo 3

```
select Student_id, file_name, image_name from image
```