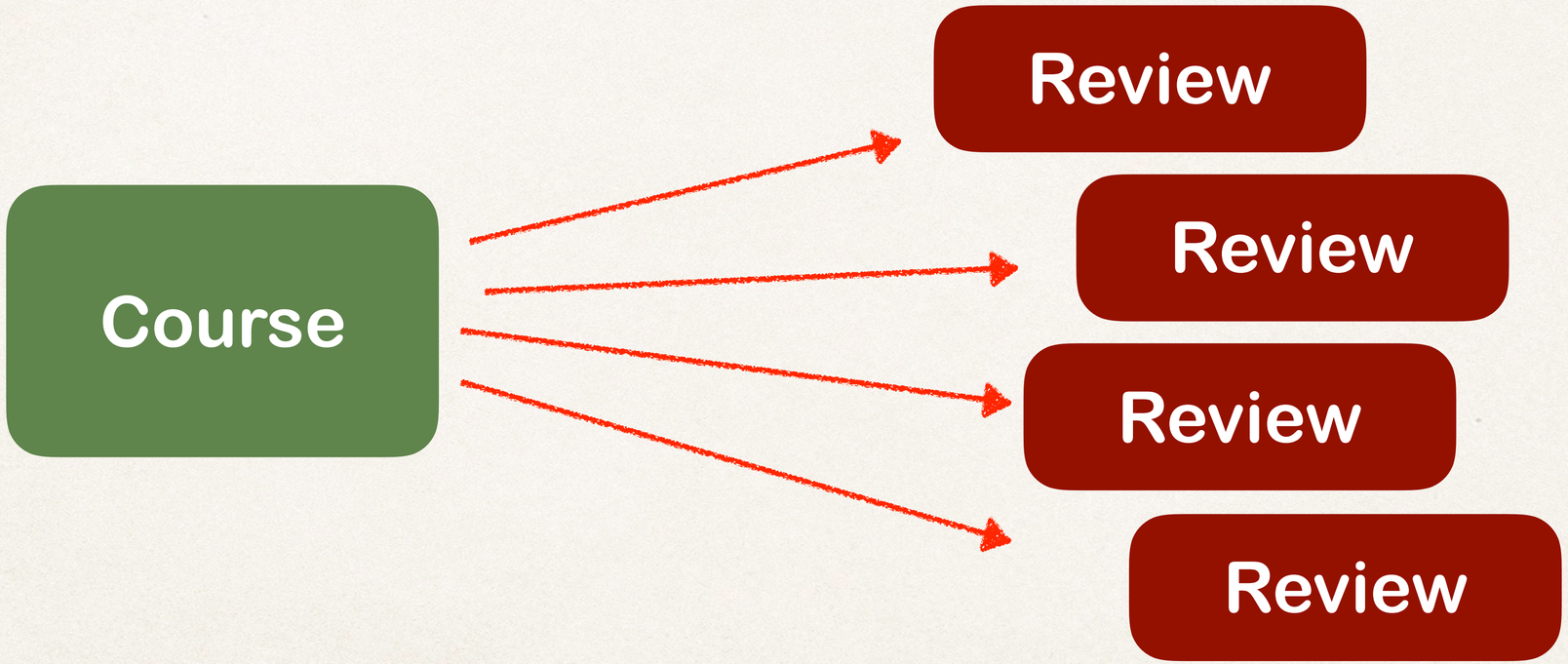
luv2code

Hibernate One-to-Many Uni-Directional

One-to-Many Mapping

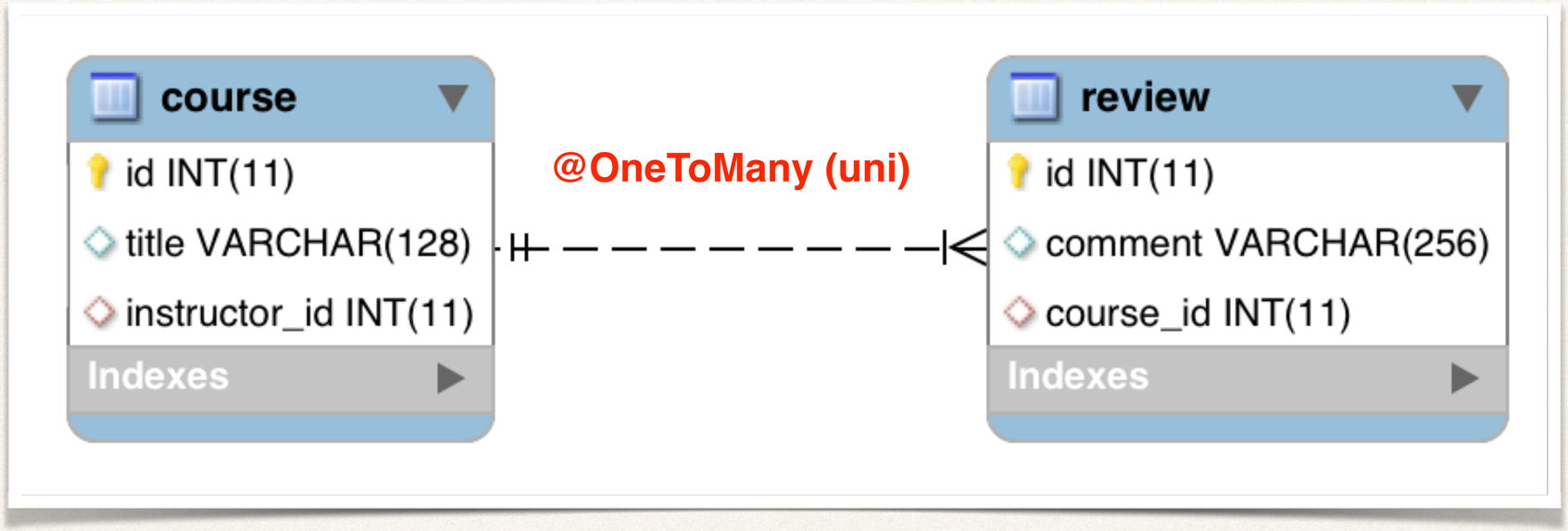
- A course can have many reviews
  - Uni-directional



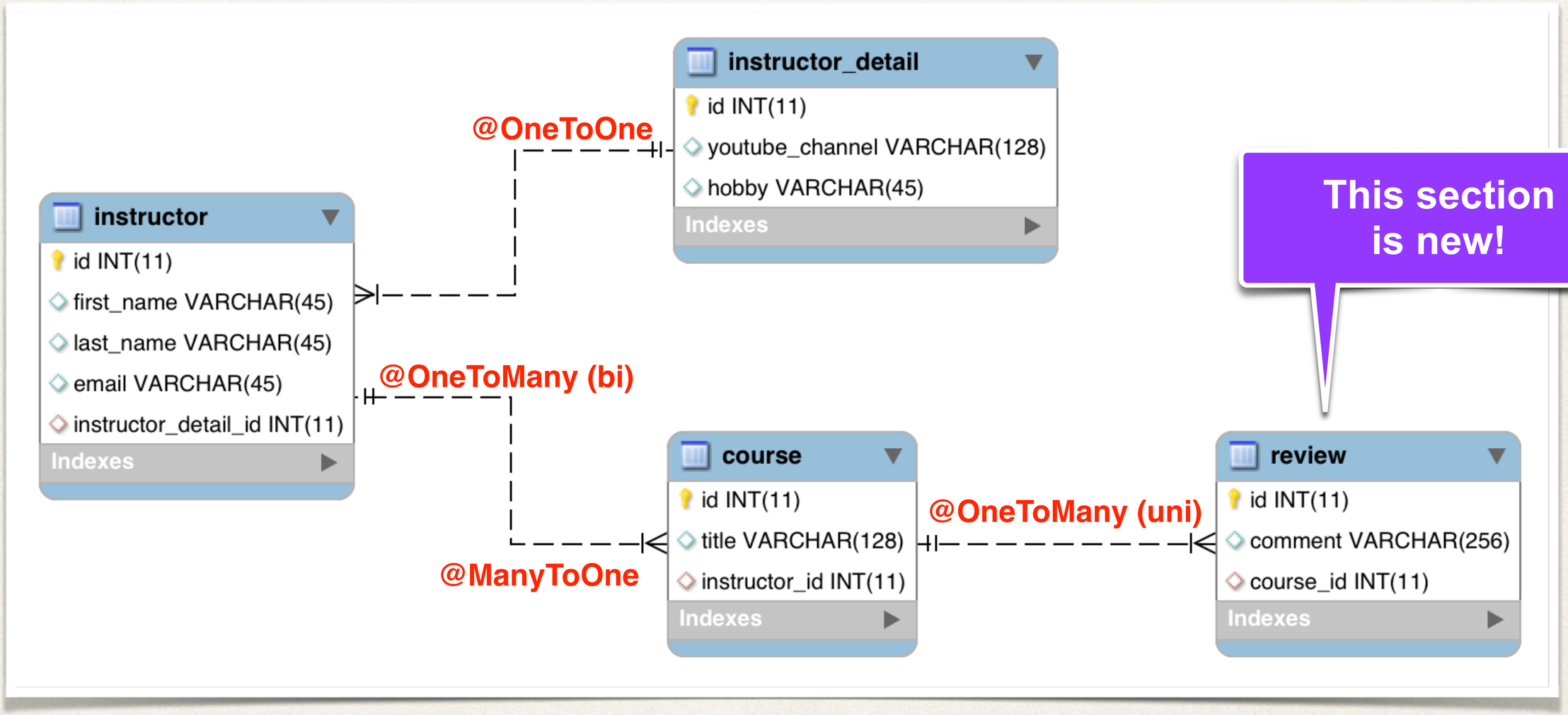
Real-World Project Requirement

- If you delete a course, also delete the reviews
- Reviews without a course … have no meaning

@OneToMany


course
id INT(11)
title VARCHAR(128)
instructor_id INT(11)
Indexes
@OneToMany (uni)
review
id INT(11)
comment VARCHAR(256)
course_id INT(11)
Indexes

Look Mom … our project is growing!



Development Process: One-to-Many

1. Prep Work - Define database tables
2. Create **Review** class
3. Update **Course** class
4. Create Main App

table: review

File: create-db.sql

```
CREATE TABLE `review` (

  `id` int(11) NOT NULL AUTO_INCREMENT,
  `comment` varchar(256) DEFAULT NULL,
  `course_id` int(11) DEFAULT NULL,

    ...
);
```

comment:
"Wow … this course is awesome!"

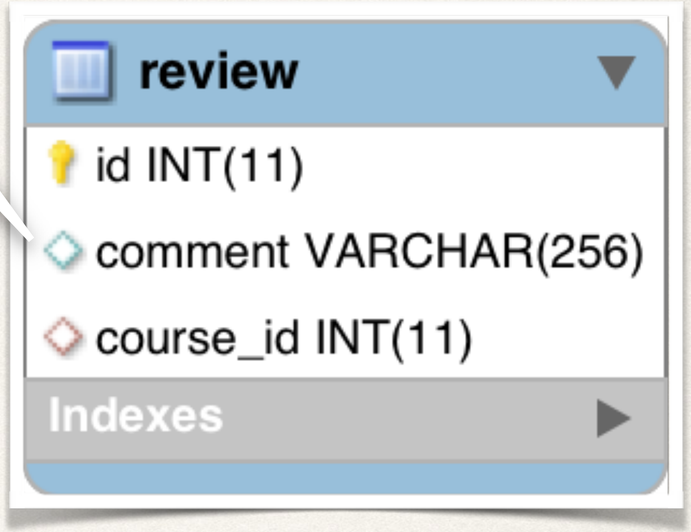


table: review - foreign key

File: create-db.sql

```
CREATE TABLE `review` (
  ...
  KEY `FK_COURSE_ID_idx` (`course_id`),
  CONSTRAINT `FK_COURSE`
  FOREIGN KEY (`course_id`)
  REFERENCES `course` (`id`)
  ...
);
```

Table

Column

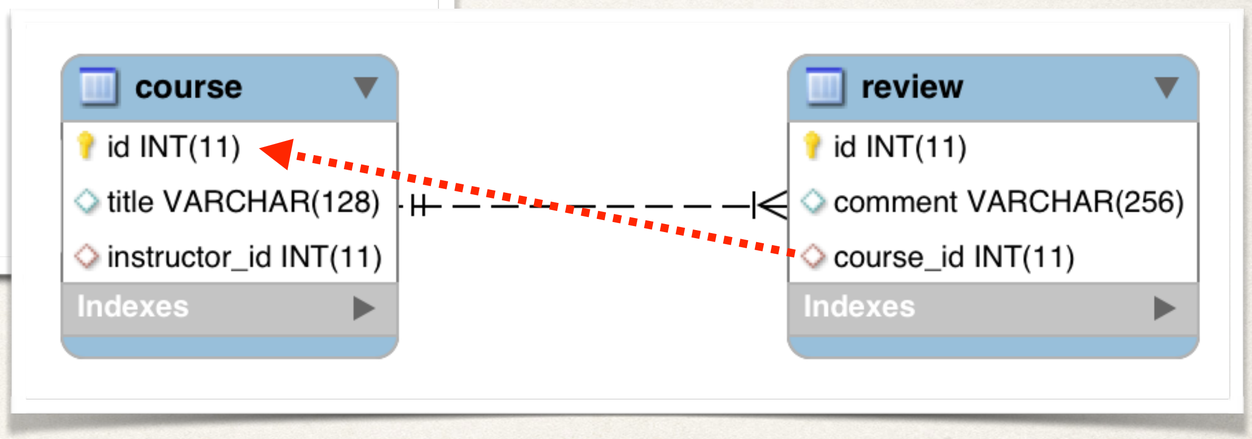
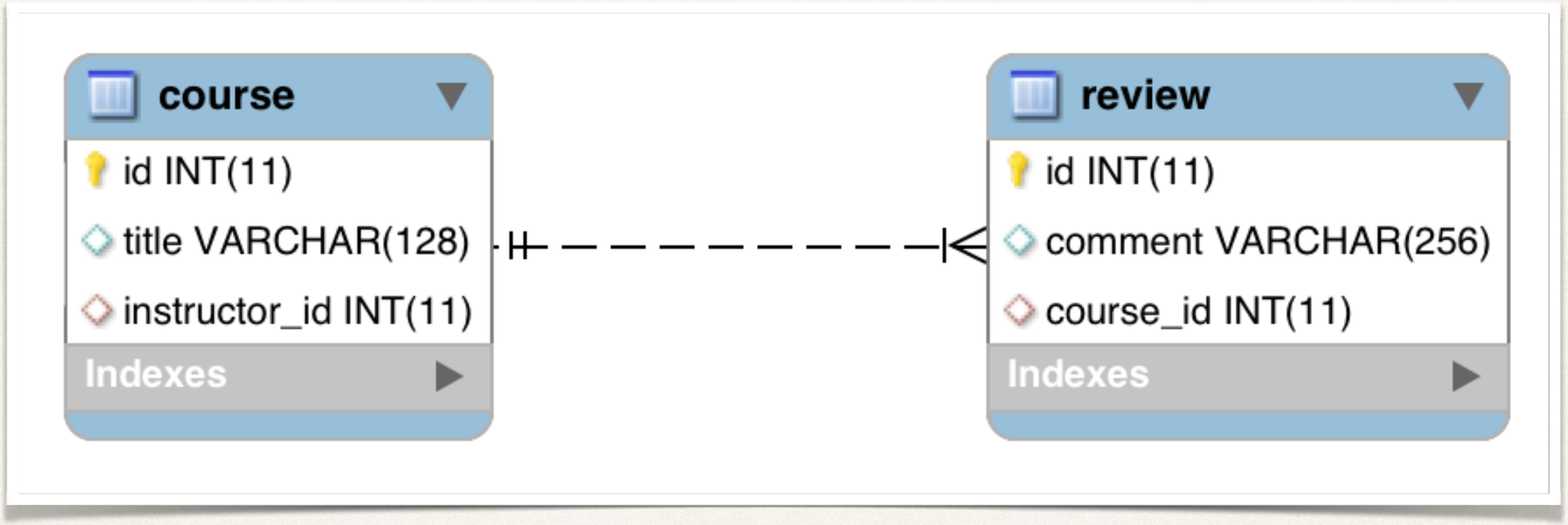
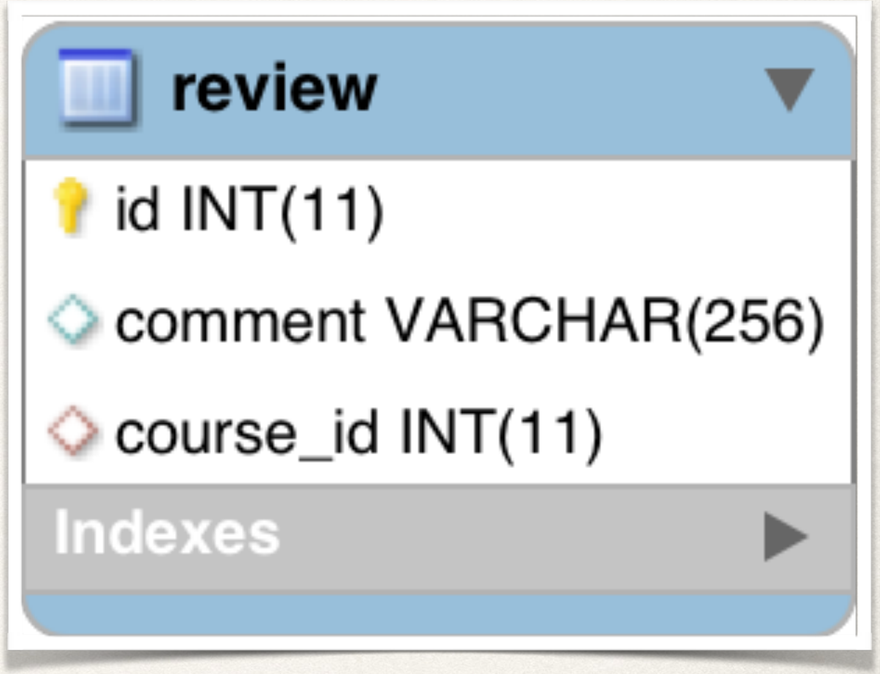


table: course - no changes



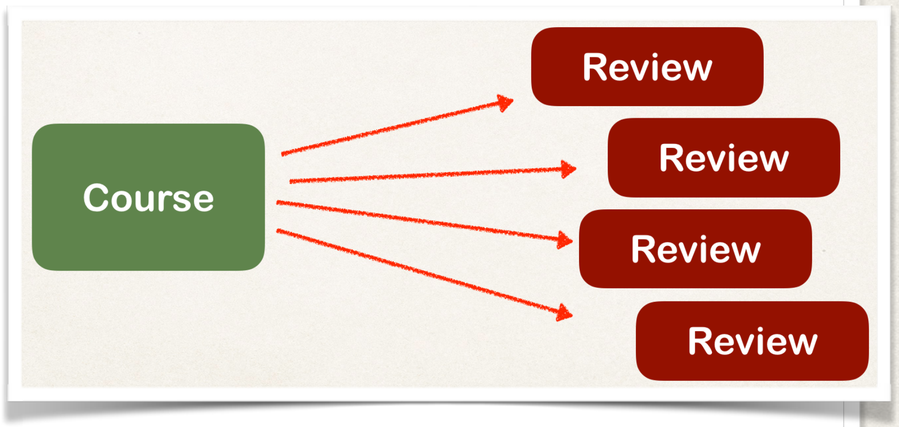
Step 2: Create Review class

```
@Entity
@Table(name="review")
public class Review {

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

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

    ...
    // constructors, getters / setters
}
```



Step 3: Update Course - reference reviews

```
@Entity
@Table(name="course")
public class Course {
    ...


    private List<Review> reviews;

    // getter / setters
    ...

}
```



Add @OneToMany annotation

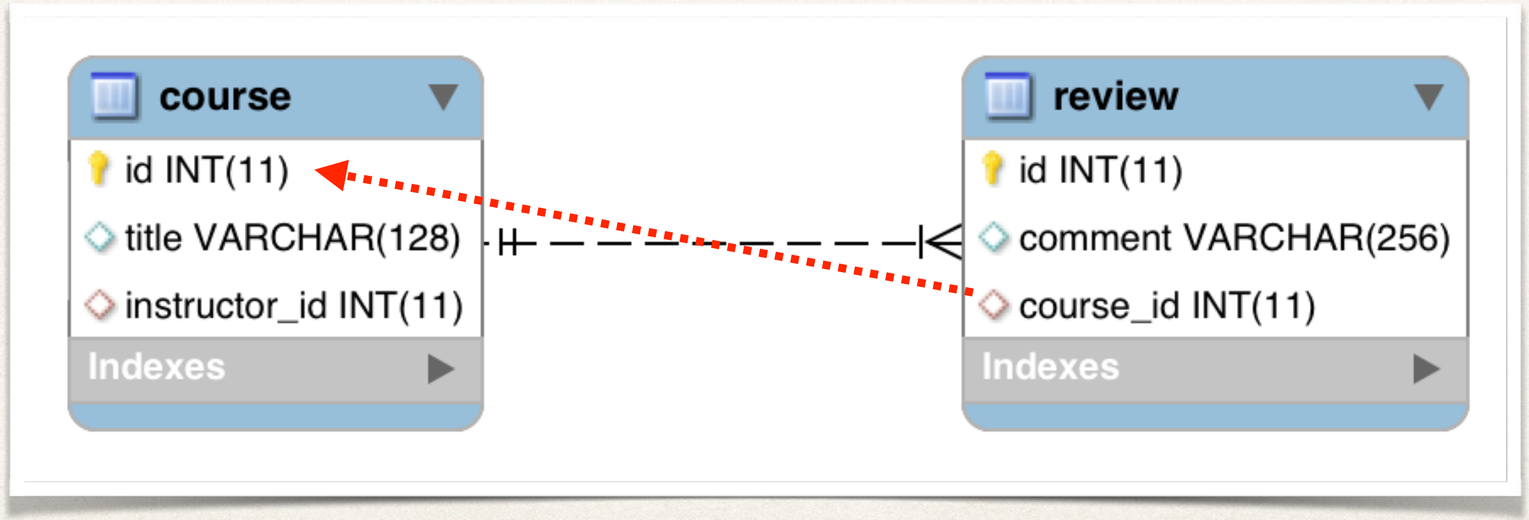
```
@Entity
@Table(name="course")
public class Course {
    ...

    @OneToMany
    @JoinColumn(name="course_id")
    private List<Review> reviews;

    // getter / setters
    ...

}
```

Refers to “course_id” column in “review” table

More: @JoinColumn

```
public class Course {
    ...

    @OneToMany
    @JoinColumn(name="course_id")
    private List<Review> reviews;
```

- In this scenario, **@JoinColumn** tells Hibernate
  - Look at the **course_id** column in the **review** table
  - Use this information to help find associated reviews for a course



Add support for Cascading

```
@Entity
@Table(name="course")
public class Course {
    ...

    @OneToMany(cascade=CascadeType.ALL)
    @JoinColumn(name="course_id")
    private List<Review> reviews;

    ...

}
```

Cascade all operations including deletes!

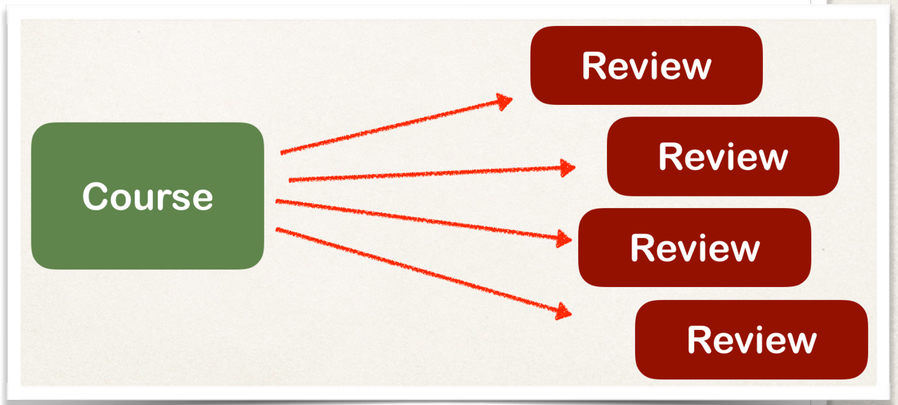
Add support for Lazy loading

Lazy load the reviews

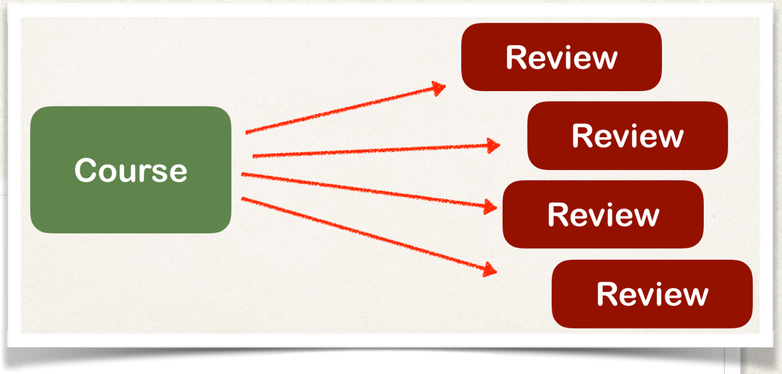
```
@Entity
@Table(name="course")
public class Course {
    ...

    @OneToMany(fetch=FetchType.LAZY, cascade=CascadeType.ALL)
    @JoinColumn(name="course_id")
    private List<Review> reviews;

    ...

}
```

Add convenience method for adding review

```
@Entity
@Table(name="course")
public class Course {
    ...
    // add convenience methods for adding reviews

    public void add(Review tempReview) {

        if (reviews == null) {
            reviews = new ArrayList<>();
        }

        reviews.add(tempReview);

    }
    ...
}
```



Step 4: Create Main App




```
public static void main(String[] args) {
        ...

        // get the course object
        int theId = 10;
        Course tempCourse = session.get(Course.class, theId);

        // print the course
        System.out.println("tempCourse: " + tempCourse);

        // print the associated reviews
        System.out.println("reviews: " + tempCourse.getReviews());
        ...
}
```

Lazy load the reviews