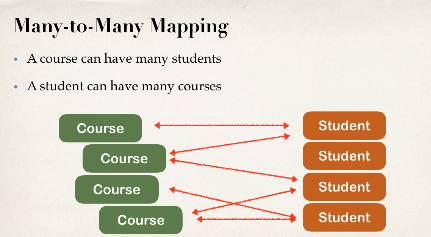
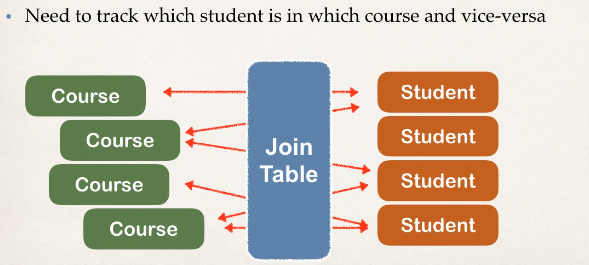
create table review (id serial PRIMARY key ,comments VARCHAR(200), course\_id int , CONSTRAINT FK\_REVIEW FOREIGN KEY (course\_id) REFERENCES course(id))

Many To Many



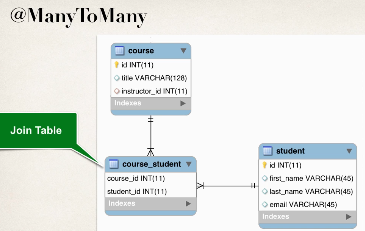
- need to track which studnet is in which course

Make use of join table



Join Table

Table that provide mapping between two tables , it has a foreign key for each table to define mapping relationship



Join table examples

Develeopment Process

1 define table

2 update couse table

3 update studnet table

4 main app

Join Table : course\_student;

CREATE table course\_student(course\_id int , student\_id int , PRIMARY KEY(course\_id,student\_id))

Join Table: course\_student - foreign keys

CREATE table course\_student(course\_id int , student\_id int , PRIMARY KEY(course\_id,student\_id),

CONSTRAINT FK\_COURSE\_STUDENT FOREIGN KEY (course\_id) REFERENCES course(id),

CONSTRAINT FK\_STUDENT\_COURSE FOREIGN KEY (student\_id) REFERENCES student(id)

)

2 update couse Entity

@ManyToMany(fetch=FetchType.***LAZY***,cascade= {CascadeType.***DETACH***,CascadeType.***MERGE***,CascadeType.***PERSIST***,CascadeType.***REFRESH***})

@JoinTable(name="course\_student",

joinColumns=@JoinColumn(name="course\_id"), // refere to course\_id column in course\_student join table

inverseJoinColumns=@JoinColumn(name="student\_id") // refere to student\_id column in course\_student join table

)

**private** List<Student> students;

// gettter and setter

// add convein method to add students

**public** **void** add(Student student)

{

**if**(students==**null**)

{

students=**new** ArrayList<Student>();

}

students.add(student);

}

@JoinTable

1 look at the course\_id column in course\_student join table

2 for the other side(inverse) look the student\_id column in the course\_student table

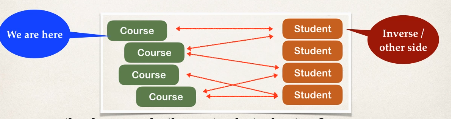
Use the information to find relation between course and student

Inverse:

In context defining relationship in the course class

The student class in another side so it is consdiered inverse

Invers refer to other side of the relationship



Lets do same way update student

3 Update Student

@ManyToMany(Student)

@JoinTable(name="course\_student",

joinColumns=@JoinColumn(name="student\_id"), // refere to student\_id column in course\_student join table

inverseJoinColumns=@JoinColumn(name="course\_id") // refere to course\_id column in course\_student join table

)

**private** List<Course> courses;

// gettet and setter

4 main APp

Real World porject

-If delete a course don’t delete student NO CASCADE DELETE

-lazy loadign of student classs

- cascading saves but not deletes

select courses0\_.student\_id as student\_1\_5\_0\_, courses0\_.course\_id as course\_i2\_5\_0\_, course1\_.id as id1\_0\_1\_, course1\_.instructor\_id as instruct3\_0\_1\_, course1\_.title as title2\_0\_1\_, instructor2\_.id as id1\_1\_2\_, instructor2\_.email as email2\_1\_2\_, instructor2\_.first\_name as first\_na3\_1\_2\_, instructor2\_.instructor\_detail\_id as instruct5\_1\_2\_, instructor2\_.last\_name as last\_nam4\_1\_2\_, instructor3\_.id as id1\_2\_3\_, instructor3\_.hobby as hobby2\_2\_3\_, instructor3\_.youtube\_channel as youtube\_3\_2\_3\_ from course\_student courses0\_ inner join basicdb.course course1\_ on courses0\_.course\_id=course1\_.id left outer join basicdb.instructor instructor2\_ on course1\_.instructor\_id=instructor2\_.id left outer join basicdb.instructor\_detail instructor3\_ on instructor2\_.instructor\_detail\_id=instructor3\_.id where courses0\_.student\_id=5

Delete Course

- confirm student not deleted

-only delete course\_student relationship

Delete Student

- confirm Course not deleted

-only delete course\_student relationship