

Unidirectional one-to-one mapping:

```
@Entity
public class Student {

    @Id
    private long studentId;

    3 usages
    @Column(name = "student_name")
    private String studentName;

    2 usages
    @OneToOne(cascade = CascadeType.ALL, fetch = FetchType.EAGER)
    @JoinColumn(name = "Student_fk")
    private StudentDetails studentDetails;

    no usages
    public long getStudentId() { return studentId; }

    2 usages
    public void setStudentId(long studentId) { this.studentId = studentId; }
```

```
StudentDetails sd1 = new StudentDetails();
sd1.setStudentDetailsId(101L);
sd1.setZipCode(393002);

StudentDetails sd2 = new StudentDetails();
sd2.setStudentDetailsId(103L);
sd2.setZipCode(393001);

Student s1 = new Student();
s1.setStudentId(1);
s1.setStudentName("Hetvi");
s1.setStudentDetails(sd2);

Student s2 = new Student();
s2.setStudentId(2);
s2.setStudentName("Aneri");
s2.setStudentDetails(sd1);
```

Bidirectional one-to-one mapping:

```
@Entity
public class StudentDetails {
    @Id
    private Long studentDetailsId;
    2 usages
    @Column(name = "zip_code")
    private int zipCode;
    2 usages
    @OneToOne(cascade = CascadeType.ALL)
    private Student student;

    no usages
    public Student getStudent() {
        return student;
    }

    2 usages
    public void setStudent(Student student) {
        this.student = student;
    }
}
```

Query 1 student studentdetails

1 • SELECT * FROM test.student;

Result Grid Filter Rows: Edit

	Student_fk	studentId	student_name
▶	103	1	Hetvi
	101	2	Aneri
*	NULL	NULL	NULL

```

StudentDetails sd1 = new StudentDetails();
sd1.setStudentDetailsId(101L);
sd1.setZipCode(393002);

StudentDetails sd2 = new StudentDetails();
sd2.setStudentDetailsId(103L);
sd2.setZipCode(393001);

Student s1 = new Student();
s1.setStudentId(1);
s1.setStudentName("Hetvi");
s1.setStudentDetails(sd2);

Student s2 = new Student();
s2.setStudentId(2);
s2.setStudentName("Aneri");
s2.setStudentDetails(sd1);

sd1.setStudent(s2);
sd2.setStudent(s1);

```

Query 1 student studentdetails x

1 • **SELECT * FROM test.studentdetails;**

Result Grid Filter Rows: Edit:

	zip_code	studentDetailsId	student_studentId
▶	393002	101	2
	393001	103	1
*	NULL	NULL	NULL

Many-to-one: (Many students are in one college)

```
@Entity
public class Student {
    @Id
    @Column(name = "student_id")
    private int studentId;

    2 usages
    @Column(name = "student_name")
    private String studentName;

    2 usages
    @ManyToOne(cascade = CascadeType.ALL)
    @JoinColumn(name = "college_fk")
    private Student_College studentCollege;

    no usages
    public Student_College getStudentCollege() { return studentCollege; }

    4 usages
    public void setStudentCollege(Student_College studentCollege) {
        this.studentCollege = studentCollege;
    }
}
```

One-to-many: (one college have many students)

```
@Entity
public class Student_College {

    @Id
    @Column(name = "college_id")
    private int collegeId;

    2 usages
    @Column(name = "college_name")
    private String collegeName;

    2 usages
    @OneToMany(mappedBy = "studentCollege", cascade = CascadeType.ALL, fetch = FetchType.EAGER)
    private List<Student> students;

    no usages
    public List<Student> getStudents() { return students; }

    2 usages
    public void setStudents(List<Student> students) { this.students = students; }
```

```
student1.setStudentCollege(college1);
student2.setStudentCollege(college2);
student3.setStudentCollege(college1);
student4.setStudentCollege(college2);
```

```
List<Student> ddit_students = new ArrayList<>();
ddit_students.add(student1);
ddit_students.add(student3);
```

```
List<Student> nirma_students = new ArrayList<>();
nirma_students.add(student2);
nirma_students.add(student4);
```

```
college1.setStudents(ddit_students);
college2.setStudents(nirma_students);
```

Query 1 student student_college

1 • `SELECT * FROM test.student;`

Result Grid Filter Rows:

	college_fk	student_id	student_name
▶	101	1	Hetvi
	102	2	Aneri
	101	3	Manisha
	102	4	Krina
*	NULL	NULL	NULL

Query 1 student student_college

1 • `SELECT * FROM test.student_college`

Result Grid Filter Rows:

	college_id	college_name
▶	101	DDIT
	102	NIRMA
*	NULL	NULL

Many-to-many: (many students have many certificates)

Create certificate entity class with id and name

```
@Entity
public class Student {
    @Id
    @Column(name = "student_id")
    private int studentId;

    2 usages
    @Column(name = "student_name")
    private String studentName;

    2 usages
    @ManyToMany(cascade = CascadeType.ALL)
    private Set<Certification> studentCertificates;

    no usages
    public Set<Certification> getStudentCertificates() {
        return studentCertificates;
    }

    4 usages
    public void setStudentCertificates(Set<Certification> studentCertificates) {
        this.studentCertificates = studentCertificates;
    }
}
```

Query 1 certification **student** student_certification

Limit to 1000 rows

```
1 • SELECT * FROM test.student;
```

Result Grid Filter Rows: Edit:

	student_id	student_name
▶	1	Hetvi
	2	Aneri
	3	Manisha
	4	Krina
*	NULL	NULL

Query 1 certification student **student_certification**

Limit to 1000 rows

```
1 • SELECT * FROM test.certification;
```

Result Grid Filter Rows: Edit:











	certification_id	certification_name
▶	101	AWS
	102	Orade
	103	Machine Learning
*	NULL	NULL

Query 1

certification

student


student_certification ×







Limit to 1000 rows ▼

1 •

`SELECT * FROM test.student_certification;`

Result Grid 

 Filter Rows:

Edit:    Exp

	Student_student_id	studentCertificates_certification_id
▶	1	101
	3	101
	4	101
	2	102
	3	102
	4	102
	1	103
	4	103