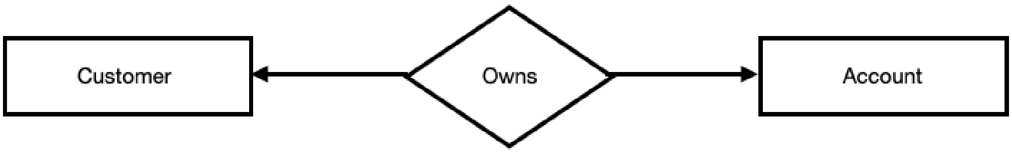
Types of Relationship

- One to One
- One to many
- Many to Many

One to One

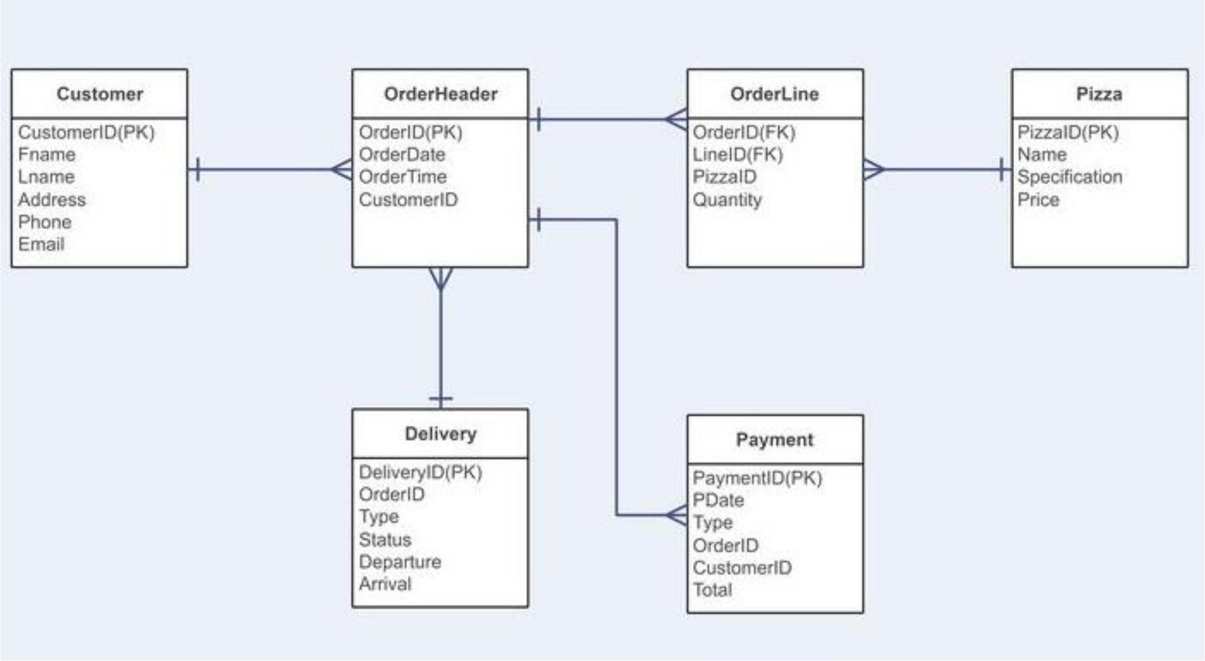
A One-to-One relationship represents a unique connection between two tables where each record appears only once in both tables.



One to One Relationship

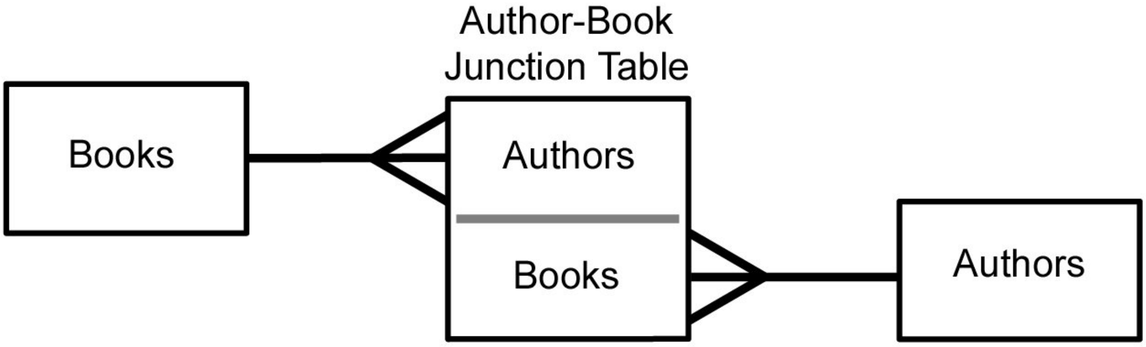
One to Many

When each entry in one table may be linked to one or more records in the other table, this is known as a one-to-many relationship.



Many to Many

A many to many relationship exists when one or more items in one table can have a relationship to one or more items in another table.



Use case of One to Many relationship

cust_id	name	email
1	Mario	mario@example.com
2	Luigi	luigi@example.com
3	Shaun	shaun@example.com

Customer

order_id	date	amount	cust_id
o1	2022-05-12	500	1
o2	2022-06-18	600	2
o3	2022-05-22	300	3

Order

FOREIGN KEY

The FOREIGN KEY constraint is used to prevent actions that would destroy links between tables. A FOREIGN KEY is a field (or collection of fields) in one table, that refers to the PRIMARY KEY in another table. The table with the foreign key is called the child table, and the table with the primary key is called the referenced or parent table.

Primary key

cust_id	name	email
1	Mario	mario@example.com
2	Luigi	luigi@example.com
3	Shaun	shaun@example.com

Customer

Primary key

Foreign Key

order_id	date	amount	cust_id
o1	2022-05-12	500	1
o2	2022-06-18	600	2
o3	2022-05-22	300	3

Order

```
CREATE TABLE customer (
    cust_id INT AUTO_INCREMENT PRIMARY KEY,
    name VARCHAR(50) NOT NULL,
    email VARCHAR(50) NOT NULL
);

CREATE TABLE orders (
    order_id INT PRIMARY KEY AUTO_INCREMENT,
    date DATE,
    amount DECIMAL(10, 2),
    cust_id INT,
    FOREIGN KEY (cust_id) REFERENCES customer(cust_id)
);
```

Checking the foreign key is set or not

```
SELECT CONSTRAINT_NAME, COLUMN_NAME, REFERENCED_TABLE_NAME FROM
INFORMATION_SCHEMA.KEY_COLUMN_USAGE WHERE TABLE_NAME='orders';
```

```
INSERT INTO customer(name, email) VALUES ('Mario', 'mario@example.com'),('Luigi',
'luigi@example.com'),('Shaun', 'shaun@example.com');
```

```
INSERT INTO orders (date, amount, cust_id) VALUES ('2022-05-22', 500.50, 1),
('2022-06-18', 600.50, 2), ('2022-05-22', 300.70, 3), ('2022-05-22', 900.70, 3), ('2022-05-22',
400.70, 2);
```

Use case of Many to Many relationship

To create a many-to-many relationship between the students and courses tables, we typically introduce a third table, often referred to as a "junction" or "pivot" table. This table will contain foreign keys that reference the primary keys of both the students and courses tables.

Students table

```
CREATE TABLE students (
   student_id INT PRIMARY KEY AUTO_INCREMENT,
   student_name VARCHAR(50) NOT NULL
);
```

Courses table

```
CREATE TABLE courses (
   course_id INT PRIMARY KEY AUTO_INCREMENT,
   course_name VARCHAR(50) NOT NULL,
   fees INT
);
```

Junction table

```
CREATE TABLE student_courses (
   student_id INT,
   course_id INT,
   PRIMARY KEY (student_id, course_id),
   FOREIGN KEY (student_id) REFERENCES students(student_id),
   FOREIGN KEY (course_id) REFERENCES courses(course_id)
);
```

The students table contains information about individual students.
The courses table contains information about individual courses.

The student_courses table serves as the junction table, linking students to the courses they are enrolled in. It contains foreign keys referencing both the student_id and course_id from their respective tables.
The composite primary key (student_id, course_id, fees) ensures that each student-course combination is unique.
To establish a relationship between a student and a course, you would insert a row into the student_courses table with the respective student_id and course_id.

This structure allows for a many-to-many relationship, where each student can be enrolled in multiple courses, and each course can have multiple students enrolled.

```
INSERT INTO students (student_name) VALUES ('Alice'),('Bob'),('Charlie'),('David'),('Eve');
INSERT INTO courses (course_name, fees) VALUES ('Java', 10000),('JavaScript',
9000),('Python', 8000),('C++',9000),('PHP', 6000);
INSERT INTO student_courses (student_id, course_id) VALUES (1, 1), (1, 3), (2, 2), (3, 3), (3,
4), (4, 1), (4, 5);
```

Let's perform join operation

```
SELECT student_name, course_name FROM students JOIN student_courses ON
student_courses.student_id=students.student_id JOIN courses ON
student_courses.course_id=courses.course_id;
```

Print No of students for each course.

```
SELECT course_name, COUNT(student_name)
FROM students
JOIN student_courses ON student_courses.student_id = students.student_id
JOIN courses ON student_courses.course_id = courses.course_id
GROUP BY course_name;
```

Print No of courses taken by each students.

```
SELECT student_name, COUNT(course_name)
FROM students
JOIN student_courses ON student_courses.student_id = students.student_id
JOIN courses ON student_courses.course_id = courses.course_id
GROUP BY student_name;
```

Print total fees paid by each student.

```
SELECT student_name, SUM(fees)
FROM students
JOIN student_courses ON student_courses.student_id = students.student_id
JOIN courses ON student_courses.course_id = courses.course_id
GROUP BY student_name;
```