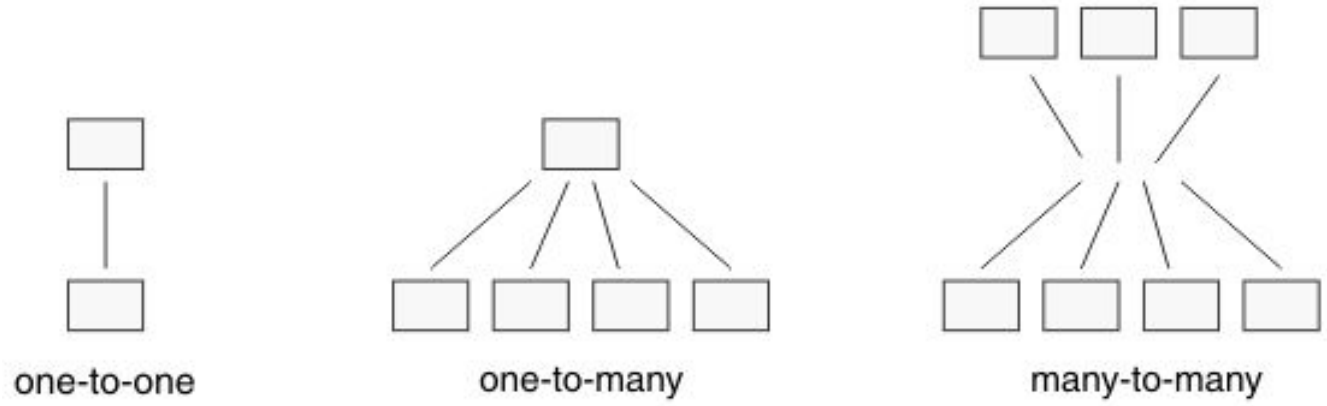
Database Relationships Types

By Aya Mostafa



INTRODUCTION

In this article, we'll talk about database relationships types.

First, why do we need relationships in databases?
Let's say that we are in our life have many relationships with others, for example, you and your mother are related. You have only one mother, but she may have several children. You may have many brothers and sisters and they also have many brothers and sisters as well. Database relationships like that, at most tables have relations between each others.

So what are the types of relationships?

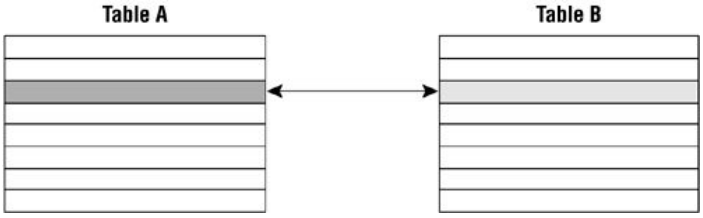
There are three types of relationships:

1. **One-to-one**
2. **One-to-many**
3. **Many-to-many**

Now we'll talk about each one in detail so let's start :)

1. ONE TO ONE

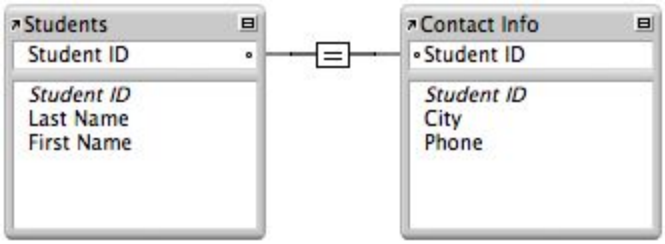
A row in table “A” can have only one matching row in table “B”, and vice versa.



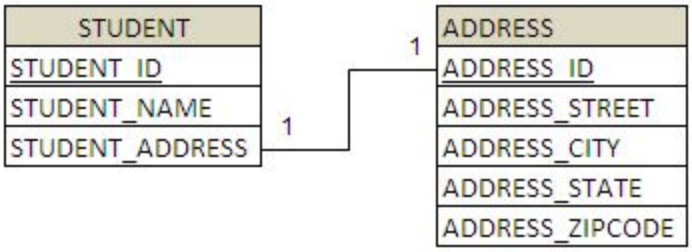
For example, in a school database, each student has only one address and each address info is assigned to only one student.

to create this relation between entities there are 3 different techniques:-

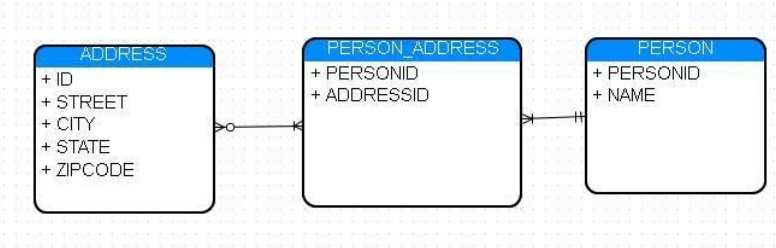
- Using shared primary key
  In this technique, we use a common primary key value in both the tables. This way primary key of entity STUDENT can safely be assumed the primary key of entity STUDENT_ADDRESS like the next figure



- Using foreign key association
  In this association, a foreign key column is created in the owner entity. For example, if we make STUDENT as owner entity, then a extra column “ADDRESS_ID” will be created in STUDENT table. This column will store the foreign key for STUDENT_ADDRESS table.



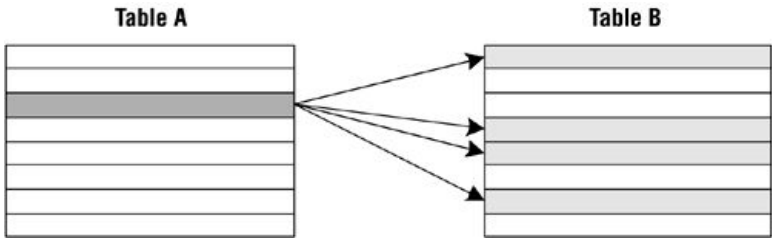
- Using a common join table
  In this association, the join table will have two foreign key columns (i.e) it will have both the primary key columns from the two entity tables (for example STUDENT_ID and ADDRESS_ID from STUDENT and STUDENT_ADDRESS entity tables respectively). One of the foreign keys will serve as the primary key for the join table. An unique constraint will be applied on the remaining foreign key column and hence both the foreign key columns will not have the duplicate values.



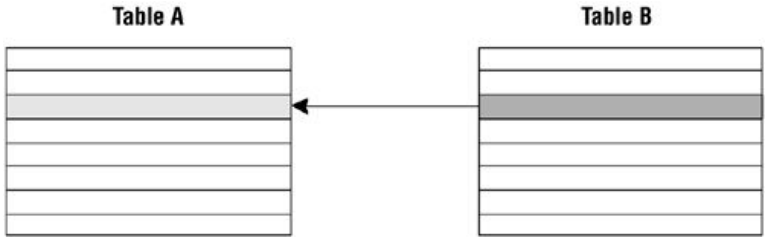
This is not a common relationship type, as the data stored in table B could just have easily been stored in table A. However, there are some valid reasons for using this relationship type. A one-to-one relationship can be used for security purposes, to divide a large table, and various other specific purposes.

2. ONE TO MANY (or MANY-TO-ONE)

This is the most common relationship type. In this type of relationship, a row in table A can have many matching rows in table B, but a row in table B can have only one matching row in table A.



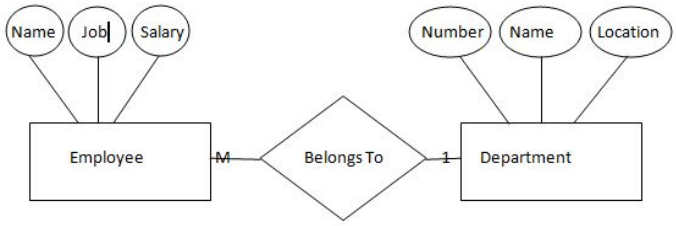
Conversely, a single record in the TABLE B can be related to only one record in TABLE A.



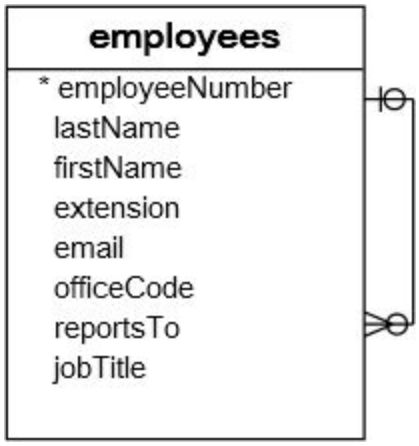
So the question is how to identify a one-to-many relationship?
When you have two entities ask yourself these questions:-

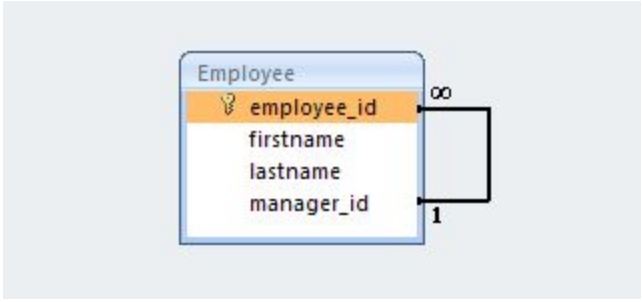
1. How many entities of B can belong to entity A?
2. How many entities of A can belong to entity B?

if the answer to question 1 is many and the answer to question 2 is one (or possibly none) you are dealing with a one-to-many relationship.
For example, one department has many employees and one employee belongs to one department



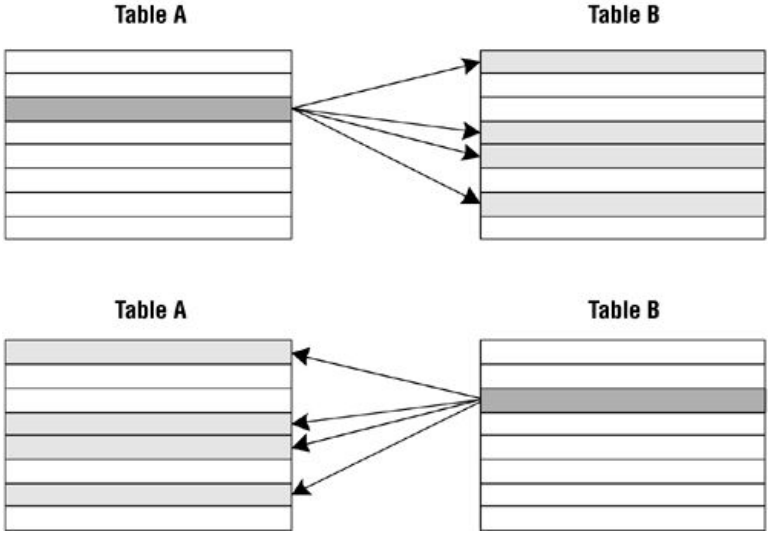
Sometimes the one to many relationship is in the same entity, for example one manager manages many employees, and each employee reports to one manager, but the manager himself is an employee also so this relationship is called “a self-Join or self-reference one-to many relationship and is represented like the next figure.





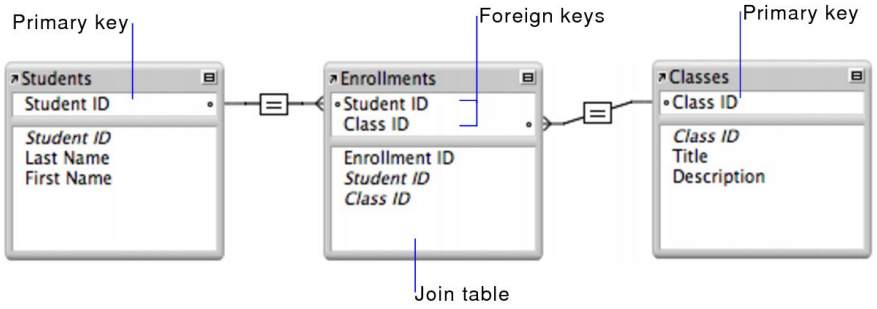
3. Many TO MANY

In a many-to-many relationship, a row in table A can have one or more records matching rows in table B, and conversely, a single record in the TABLE B can be related to one or more records in TABLE A .



This is the second most common relationship that exists between a pair of tables in a database.

For example, relationship between students and classes. A student can register for many classes, and a class can include many students so to make this type of relationship between those two entities will create third column called "**join table**". This table will take the student id and class id as foreign keys.



As there is a type of self-join one-to-many relationship, there is also self-referencing many-to-many relationship exists when a given record in the table can be related to one or more other records within the table and one or more records can themselves be related to the given record.For example, in the next figure a particular part can comprise several different component parts, and it can itself be a component of other parts. As (Part ID 704) is composed of (Part ID 703), (Part ID 702), (Part ID 701). Additionally, the (Part ID 704) is itself a component of (Part ID 707) and (Part ID 711).

Parts

Part ID	Part Name	*<< other fields >>*
701	Top Clamp	.....
702	Bottom Clamp	.....
703	Fastening Bolt	.....
704	Clamp Assembly	.....
705	Saddle	.....
706	Seatpost	.....
707	Seat Assembly	.....
708	Body Tube	.....
709	Front Fork Tube	.....
710	Rear Stay Tube	.....
711	Frame Assembly	.....

Conclusion

By the end of this article, we talked about the main database relationships types and in the next article we'll see how can we implement those relationships in laravel which provides a great feature called "Eloquent relationships" that makes managing and working with these relationships easy, and supports several different types of relationships.