DOT NET TUTORIALS

Introduction & Environment Setup

- Introduction to SQL
- How to Download and Install MySQL on Windows OS
- An Overview of MySQL Database
- Working with MYSQL Workbench
- Create new user using MySQL Workbench
- Create Alter Drop Database using MySQL Workbench
- Create Alter Drop Table using MySQL Workbench
- Insert Update Delete Data Rows using MySQL Workbench
- Database Export and Import Using MySQL Workbench
- Data Types in MySQL

MySQL Clauses

- SELECT Statement in MySQL
- WHERE Clause in MySQL
- GROUP BY Clause in MySQL
- Having Clause in MySQL
- Order by Clause in MySQL
- LIMIT Clause in MySQL
- Alias in MySQL

Conditions & Operators in MySQL

- Conditions in MySQL
- Comparison Operators in MySQL
- Logical AND, OR, NOT Operators in MySQL
- LIKE Operator in MySQL
- IN and BETWEEN Operator in MySQL
- EXISTS Operator in MySQL
- Arithmetic Operators in MySQL

Database Relationships in MySQL

Back to: MySQL Tutorials for Beginners and Professionals

Database Relationships in MySQL with Examples

In this article, I am going to discuss **Database Relationships in MySQL** with Examples. Please read our previous article where we discussed **SQL Injection in MySQL**. At the end of this article, you will understand everything about the One-to-One, One-to-Many, and Many-to-Many Database relationships in MySQL with Examples.

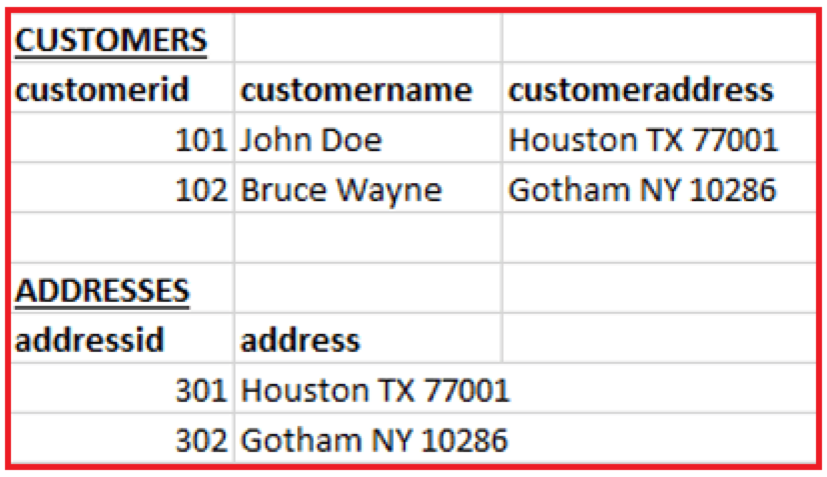
Database Relationships in MySQL

Database relationship means how the data in one table is related to the data in another table. In RDBMS (Relational Database Management System). The term "Relational" refers to the tables with Relations. Relationships between two tables are created using keys. A key in one table will normally relate to a key in another table. Two tables in a database may also be unrelated. There are mainly 3 types of database relationships:

1. **One-to-one (1:1) Relationship:** If only one data in one table relates to the only one data in another table it is known as one-to-one (1:1) relationship.
2. **One-to-many (1:M) Relationship:** If only one data in one table relates to the multiple data in another table it is known as the one-to-many (1:M) relationship.
3. **Many-to-many (M:M) Relationship:** And if multiple data in one table relates to the multiple data in another table it is known as many-to-many (M:M) relationship.

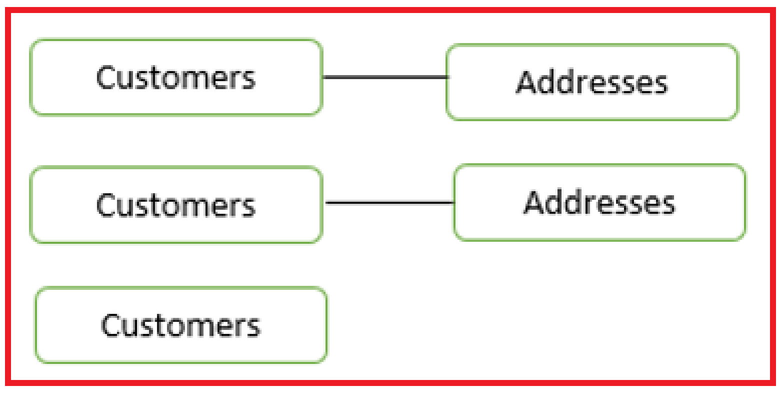
One-to-One (1:1) Database Relationship in MySQL

In a One-to-One (1:1) Relationship only one data in one table relates to only one data in another table. Take a look at the example tables. We have two database tables i.e. customers and addresses. First table stores customerid, customername, and addressid. The second table stores addressid and address column values which are shown in the below image.



CUSTOMERS		
customerid	customername	customeraddress
101	John Doe	Houston TX 77001
102	Bruce Wayne	Gotham NY 10286
ADDRESSES		
addressid	address	
301	Houston TX 77001	
302	Gotham NY 10286	

In customers table, each addressid in a data row represents the actual address of the customer in the address table. Every customer can have only one address and everyone address can have only one customer. This represents One-to-One Relationships between the two tables.



- Concatenation and Temporal Operators in MySQL
- SET Operators (UNION, UNION ALL, INTERSECT, & EXCEPT) in MySQL
- Assignment Operator in MySQL

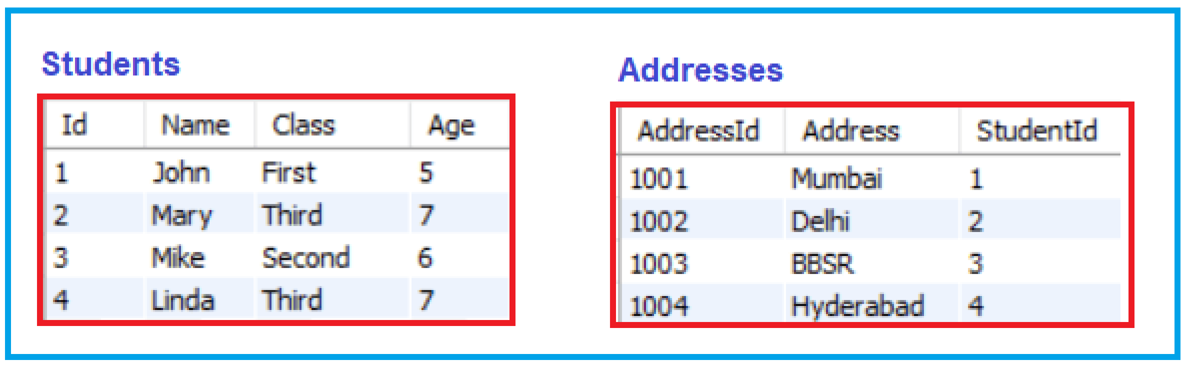
MySQL – Constraints
- Constraints in MySQL
- NOT NULL Constraint in MySQL
- Default Constraint in MySQL
- UNIQUE Constraint in MySQL
- Check Constraints in MySQL
- PRIMARY Key Constraint in MySQL
- Foreign Key Constraint in MySQL
- Referential Integrity Constraints in MySQL

MySQL – Built-in Functions
- COUNT Function in MySQL
- MySQL SUM Function
- MySQL MIN and MAX Function
- MySQL AVG Function
- MySQL UCASE and LCASE Function
- MySQL MID Function
- MySQL LENGTH and CHAR_LENGTH Function
- MySQL Round Function
- MySQL FORMAT Function
- MySQL SUBSTRING Function
- MySQL SUBSTR Function
- MySQL COALESCE Function
- MySQL CAST Function
- MySQL CONVERT Function

MySQL – JOINs
- Joins in MySQL
- Inner Join in MySQL
- Left Outer Join in MySQL
- Right Outer Join in MySQL
- Self Join in MySQL

Examples to Understand One-to-One Relationship in MySQL

Let's understand One-to-One Database Relationship in MySQL with examples. Now we will create the database school and within the school database, we will create two tables i.e. Students and Addresses as shown in the below image. Then we will see the One-To-One Relationship between Students and Addresses table.


Students

Id	Name	Class	Age
1	John	First	5
2	Mary	Third	7
3	Mike	Second	6
4	Linda	Third	7

Addresses

AddressId	Address	StudentId
1001	Mumbai	1
1002	Delhi	2
1003	BBSR	3
1004	Hyderabad	4

In the Addresses table, the column value AddressId 1001, 1002, 1003, and 1004 are assigned to the StudentId values 1, 2, 3, and 4 respectively. The details of each student are stored in Students table. In the case of these 2 tables, every student has only 1 Address while every AddressId can only be assigned to only 1 student. This type of relationship between these 2 tables is known as One-To-One Relationship.

Please use the below SQL Script to create the SCHOOL database, and create the Students and Addresses tables with the required sample data.

```sql
CREATE DATABASE SCHOOL;
USE SCHOOL;

-- Create Students Table
CREATE TABLE Students (
     Id INT PRIMARY KEY,
     Name VARCHAR(40) NOT NULL,
     Class VARCHAR(20),
     Age INT
);

-- Populate the Students Table with test data
INSERT INTO Students VALUES (1,'John', 'First', 5);
INSERT INTO Students VALUES (2, 'Mary', 'Third', 7);
INSERT INTO Students VALUES (3, 'Mike', 'Second', 6);
INSERT INTO Students VALUES (4, 'Linda', 'Third', 7);

-- Create Addresses Table
CREATE TABLE Addresses (
     AddressId INT PRIMARY KEY,
     Address VARCHAR(100) NOT NULL,
     StudentId INT NOT NULL UNIQUE
);

-- Populate the Addresses Table with test data
INSERT INTO Addresses VALUES(1001, 'Mumbai', 1);
INSERT INTO Addresses VALUES(1002, 'Delhi', 2);
INSERT INTO Addresses VALUES(1003, 'BBSR', 3);
INSERT INTO Addresses VALUES(1004, 'Hyderabad', 4);
```

One-To-Many (1:M) Database Relationship

In One-To-Many (1:M) Relationship one records in one table relate to multiple records in another table. The One-To-Many (1:M) Relationships is the most commonly used relationship in relational database management systems.

Let us understand the one-to-many relationships with an example. We have two database tables i.e. Customers and Orders. Customers table stores customerid and customername. The orders table stores orderid, customerid, orderdate and amount column values as shown in the below image.

MySQL – Window Functions

- OVER Clause in MySQL
- Row_Number Function in MySQL
- Rank Function in MySQL
- DENSE_RANK Function in MySQL

MySQL – Views

- Views in MySQL
- Advantages and Disadvantages of Views in MySQL
- Complex Views in MySQL

MySQL – Transaction Control Language

- Transaction Management in MySQL
- SAVEPOINT in MySQL

MySQL – User-Defined Functions, Procedures & Triggers

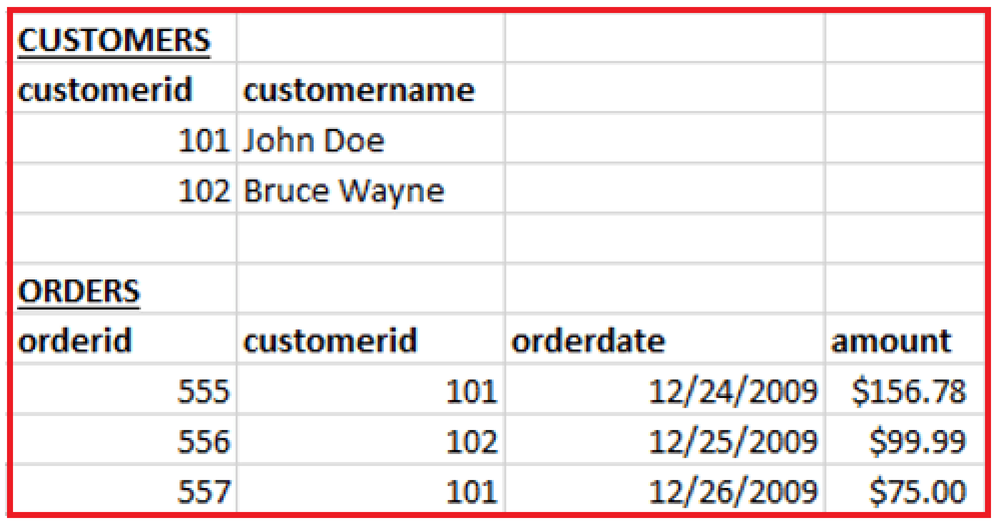
- User Defined Functions in MySQL

MySQL – Advanced Concepts

- SQL Injection in MySQL
- **Database Relationships in MySQL**
- Database Normalization in MySQL
- Database Denormalization in MySQL

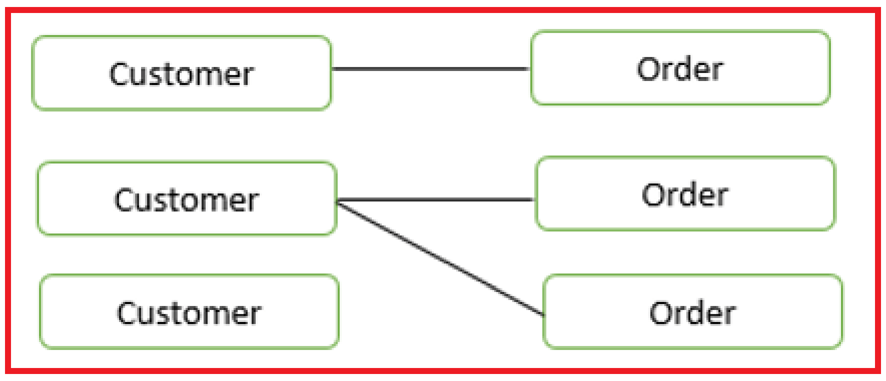
MySQL – Popular Books

- Most Recommended MySQL Books
- Most Recommended MySQL DBA Books



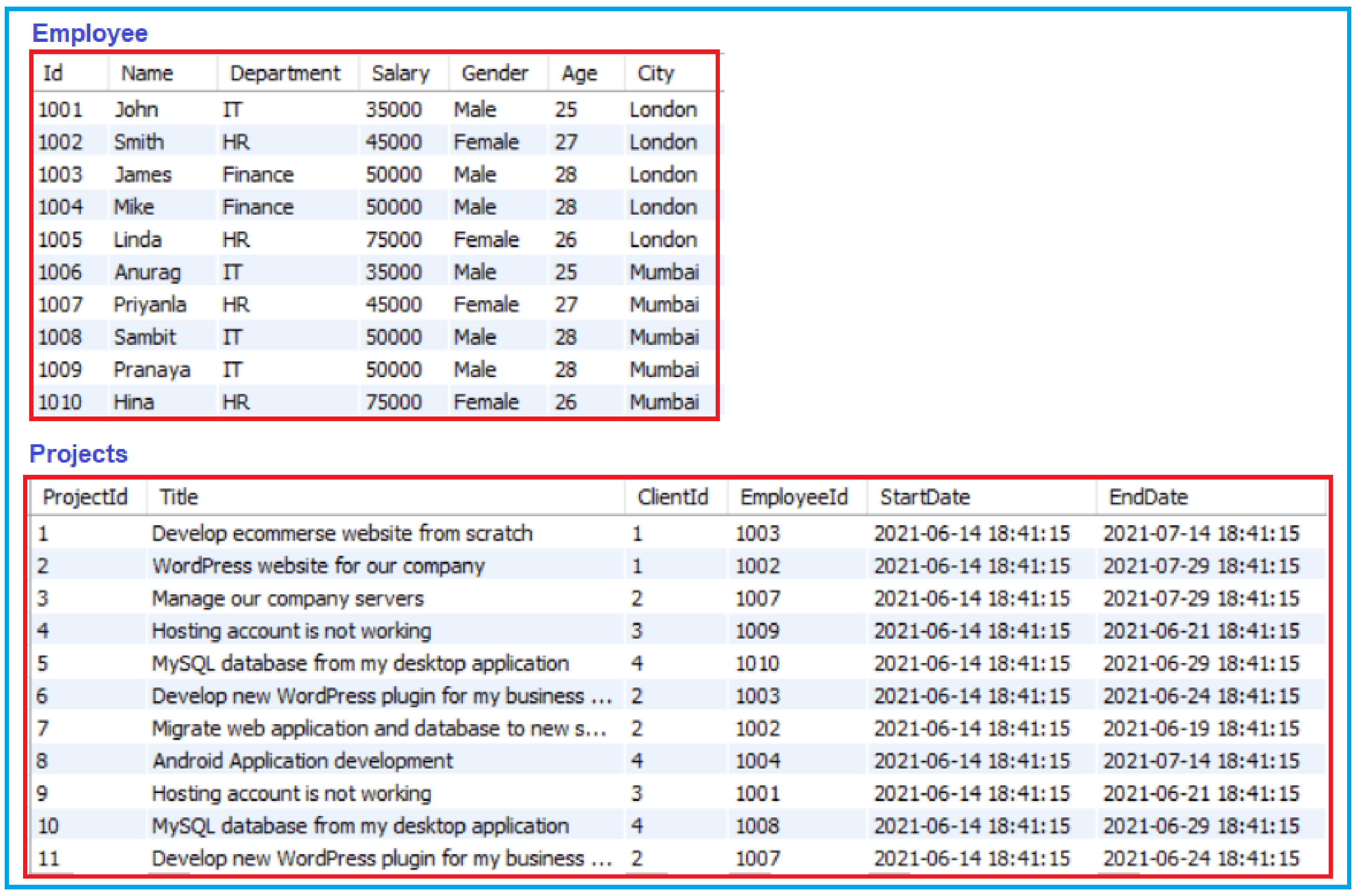
CUSTOMERS			
customerid	customername		
101	John Doe		
102	Bruce Wayne		
ORDERS			
orderid	customerid	orderdate	amount
555	101	12/24/2009	$156.78
556	102	12/25/2009	$99.99
557	101	12/26/2009	$75.00

In orders tables the customerid data row represents the customerid of a customer from customers table. A single customer from customers table can order multiple items. Therefore, the orders table can contain more than one customerid of the same customer in customerid column. But there can be only one customerid in the 'customerid' column in 'customers' table. This represents the One-To-Many Relationship between the Customers and Orders tables as shown in the below image.



Example to Understand One-To-Many (1:M) Database Relationship in MySQL

Let us understand 0ne-To-Many Relationships with examples in MySQL. Now we will create the database Company and within the Company database will create two tables i.e. Employee and Projects as shown in the below image. Then we will see the One-To-Many Relationships between Employee and Projects table.



Employee

Id	Name	Department	Salary	Gender	Age	City
1001	John	IT	35000	Male	25	London
1002	Smith	HR	45000	Female	27	London
1003	James	Finance	50000	Male	28	London
1004	Mike	Finance	50000	Male	28	London
1005	Linda	HR	75000	Female	26	London
1006	Anurag	IT	35000	Male	25	Mumbai
1007	Priyanla	HR	45000	Female	27	Mumbai
1008	Sambit	IT	50000	Male	28	Mumbai
1009	Pranaya	IT	50000	Male	28	Mumbai
1010	Hina	HR	75000	Female	26	Mumbai

Projects

ProjectId	Title	ClientId	EmployeeId	StartDate	EndDate
1	Develop ecommerse website from scratch	1	1003	2021-06-14 18:41:15	2021-07-14 18:41:15
2	WordPress website for our company	1	1002	2021-06-14 18:41:15	2021-07-29 18:41:15
3	Manage our company servers	2	1007	2021-06-14 18:41:15	2021-07-29 18:41:15
4	Hosting account is not working	3	1009	2021-06-14 18:41:15	2021-06-21 18:41:15
5	MySQL database from my desktop application	4	1010	2021-06-14 18:41:15	2021-06-29 18:41:15
6	Develop new WordPress plugin for my business ...	2	1003	2021-06-14 18:41:15	2021-06-24 18:41:15
7	Migrate web application and database to new s...	2	1002	2021-06-14 18:41:15	2021-06-19 18:41:15
8	Android Application development	4	1004	2021-06-14 18:41:15	2021-07-14 18:41:15
9	Hosting account is not working	3	1001	2021-06-14 18:41:15	2021-06-21 18:41:15
10	MySQL database from my desktop application	4	1008	2021-06-14 18:41:15	2021-06-29 18:41:15
11	Develop new WordPress plugin for my business ...	2	1007	2021-06-14 18:41:15	2021-06-24 18:41:15

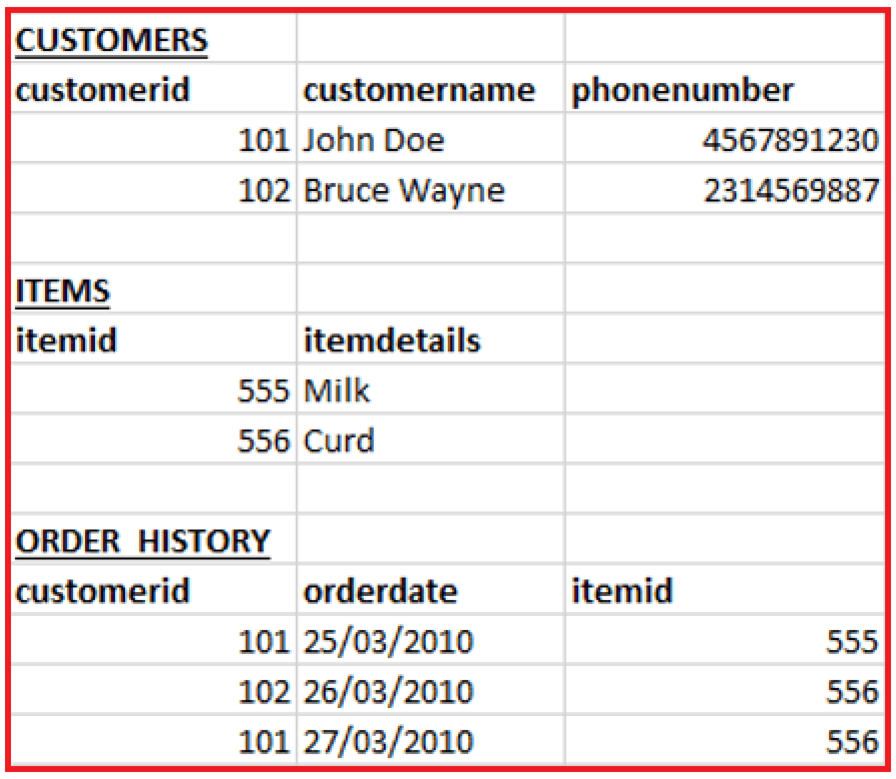
In the Projects table, one employee can post multiple projects, therefore, the Projects table may contain more than one 'employeeid' value for the same employee. But the same employee cannot have multiple employees in the id column. Take a look at the 'employeeid' 1002, 1003, and 1007 in the 'Projects' table. The clients have posted 2 projects respectively for the above three employees. And their employeeid is saved twice in the 'projects' table but there is only one employeeid for each employee in the Employee table. This represents the One-To-Many Relationship between the two tables.

Please use the below SQL Script to create the Company database, and create the Employee and Projects tables with the required sample data.

```
CREATE DATABASE Company;
USE Company;

-- Create Students Table
CREATE TABLE Employee (
 Id INT PRIMARY KEY,
 Name VARCHAR(45) NOT NULL,
 Department VARCHAR(45) NOT NULL,
 Salary FLOAT NOT NULL,
 Gender VARCHAR(45) NOT NULL,
 Age INT NOT NULL,
 City VARCHAR(45) NOT NULL
);

-- Populate the Employee Table with test data
 INSERT INTO Employee VALUES (1001, 'John', 'IT', 35000, 'Male', 25,
'London');
 INSERT INTO Employee VALUES (1002, 'Smith', 'HR', 45000, 'Female',
27, 'London');
 INSERT INTO Employee VALUES (1003, 'James', 'Finance', 50000, 'Male',
28, 'London');
 INSERT INTO Employee VALUES (1004, 'Mike', 'Finance', 50000, 'Male',
28, 'London');
 INSERT INTO Employee VALUES (1005, 'Linda', 'HR', 75000, 'Female',
26, 'London');
 INSERT INTO Employee VALUES (1006, 'Anurag', 'IT', 35000, 'Male', 25,
'Mumbai');
 INSERT INTO Employee VALUES (1007, 'Priyanla', 'HR', 45000, 'Female',
27, 'Mumbai');
 INSERT INTO Employee VALUES (1008, 'Sambit', 'IT', 50000, 'Male', 28,
'Mumbai');
 INSERT INTO Employee VALUES (1009, 'Pranaya', 'IT', 50000, 'Male',
28, 'Mumbai');
 INSERT INTO Employee VALUES (1010, 'Hina', 'HR', 75000, 'Female', 26,
'Mumbai');

-- Create Projects Table
CREATE TABLE Projects (
 ProjectId INT PRIMARY KEY,
        Title VARCHAR(200) NOT NULL,
        ClientId INT,
        EmployeeId INT,
        StartDate DATETIME,
        EndDate DATETIME,
        FOREIGN KEY (EmployeeId) REFERENCES Employee(Id)
);

-- Populate the Projects Table with test data
 INSERT INTO Projects VALUES (1, 'Develop ecommerse website from
scratch', 1, 1003, NOW(), DATE_ADD(NOW(), INTERVAL 30 DAY));
 INSERT INTO Projects VALUES (2, 'WordPress website for our company',
1, 1002, NOW(), DATE_ADD(NOW(), INTERVAL 45 DAY));
 INSERT INTO Projects VALUES (3, 'Manage our company servers', 2,
1007, NOW(), DATE_ADD(NOW(), INTERVAL 45 DAY));
 INSERT INTO Projects VALUES (4, 'Hosting account is not working', 3,
1009, NOW(), DATE_ADD(NOW(), INTERVAL 7 DAY));
 INSERT INTO Projects VALUES (5, 'MySQL database from my desktop
application', 4, 1010, NOW(), DATE_ADD(NOW(), INTERVAL 15 DAY));
 INSERT INTO Projects VALUES (6, 'Develop new WordPress plugin for my
business website', 2, 1003, NOW(), DATE_ADD(NOW(), INTERVAL 10 DAY));
 INSERT INTO Projects VALUES (7, 'Migrate web application and database
to new server', 2, 1002, NOW(), DATE_ADD(NOW(), INTERVAL 5 DAY));
 INSERT INTO Projects VALUES (8, 'Android Application development', 4,
1004, NOW(), DATE_ADD(NOW(), INTERVAL 30 DAY));
 INSERT INTO Projects VALUES (9, 'Hosting account is not working', 3,
1001, NOW(), DATE_ADD(NOW(), INTERVAL 7 DAY));
 INSERT INTO Projects VALUES (10, 'MySQL database from my desktop
application', 4, 1008, NOW(), DATE_ADD(NOW(), INTERVAL 15 DAY));
 INSERT INTO Projects VALUES (11, 'Develop new WordPress plugin for my
business website', 2, 1007, NOW(), DATE_ADD(NOW(), INTERVAL 10 DAY));
```

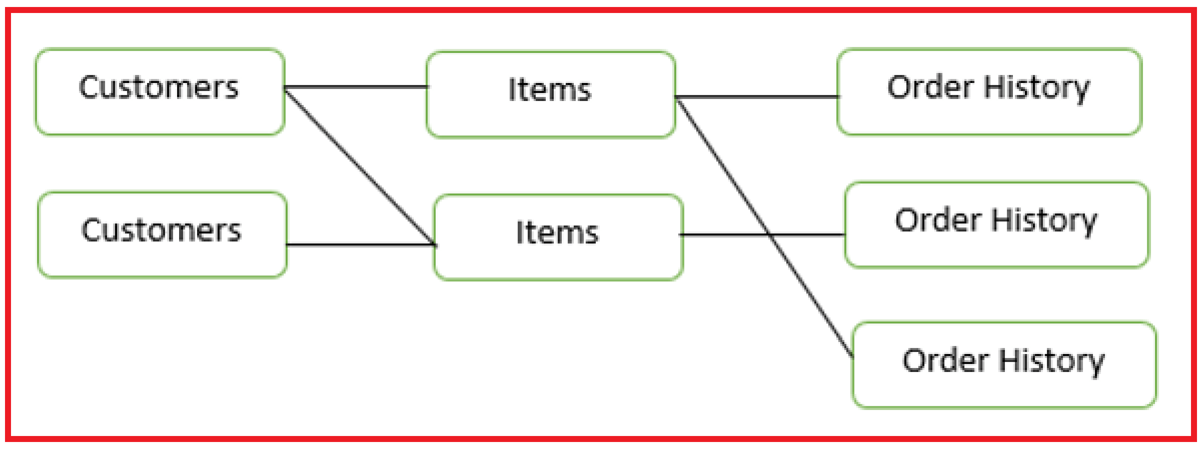
Many-To-Many (M:M) Database Relationship

In Many-To-Many (M:M) Database Relationship, multiple records in one table relate to multiple records in another table. Take a look at the below example tables. We have 3 database tables. The first table called ‘customers’, stores customerid name and phone number of the customers. The second table called ‘items’, stores itemid, and item_details, and the third table called ‘order_history’ stores, customerid, orderdate, and itemid as shown in the below image.



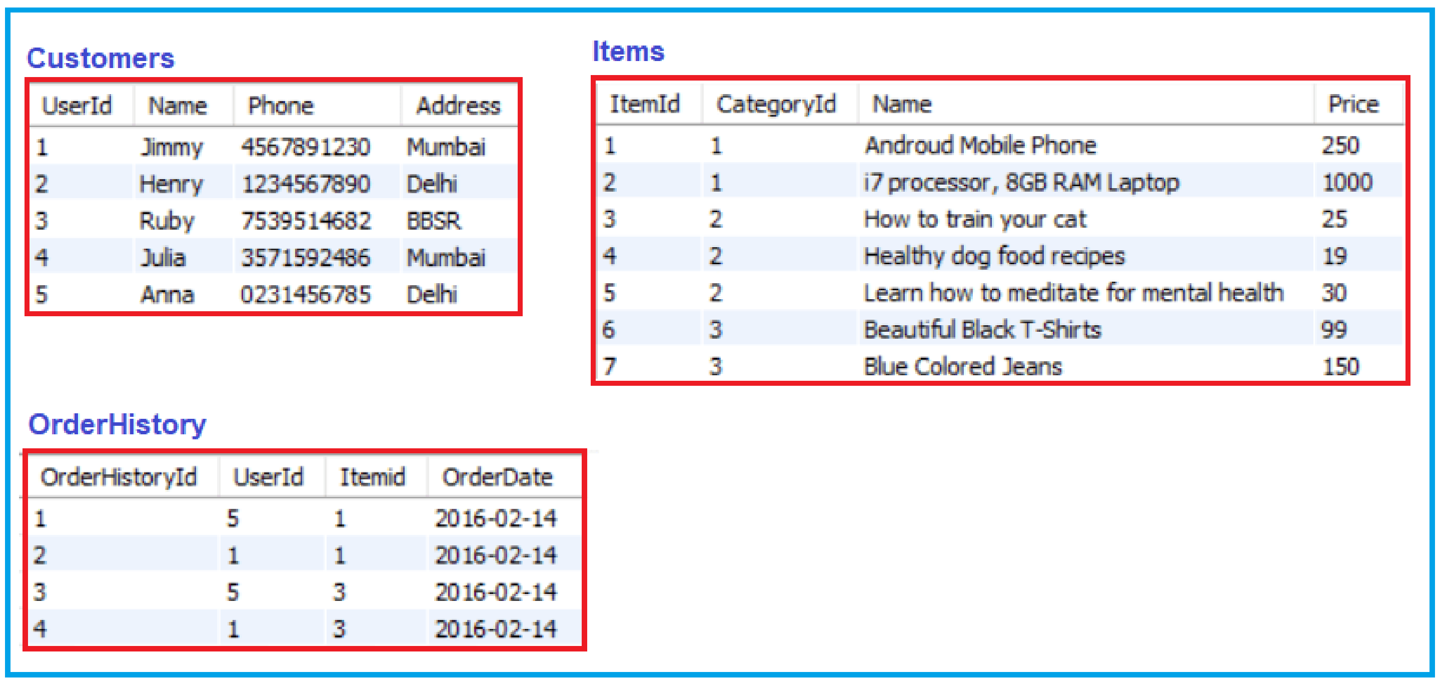
CUSTOMERS		
customerid	customername	phonenumber
101	John Doe	4567891230
102	Bruce Wayne	2314569887
ITEMS		
itemid	itemdetails	
555	Milk	
556	Curd	
ORDER_HISTORY		
customerid	orderdate	itemid
101	25/03/2010	555
102	26/03/2010	556
101	27/03/2010	556

In the OrderHistory table, the customerid data row represents the customerid of a customer from Customers table. And the itemid data row represents the itemid of an item from the items table. A single customers table can make multiple orders. Therefore, customerid column of OrderHistory table can store the same customerid multiple times. Similarly, single item from the Items table can be ordered multiple times. Therefore, itemid column of the OrderHistory table can store the same itemid multiple times. This represents Many-To-Many Relationship between the three tables as shown in the below image.



Example to Understand Many-To-Many (M:M) Database Relationship in MySQL

Let us understand Many-To-Many Database Relationships with examples in MySQL. Now we will create the database OnlineShop and within the OnlineShop database, we will create three tables i.e. Customers, Items, and OrderHistory as shown in the below image. Then we will see the Many-To-Many Relationships between these three tables.



Customers

UserId	Name	Phone	Address
1	Jimmy	4567891230	Mumbai
2	Henry	1234567890	Delhi
3	Ruby	7539514682	BBSR
4	Julia	3571592486	Mumbai
5	Anna	0231456785	Delhi

Items

ItemId	CategoryId	Name	Price
1	1	Androud Mobile Phone	250
2	1	i7 processor, 8GB RAM Laptop	1000
3	2	How to train your cat	25
4	2	Healthy dog food recipes	19
5	2	Learn how to meditate for mental health	30
6	3	Beautiful Black T-Shirts	99
7	3	Blue Colored Jeans	150

OrderHistory

OrderHistoryId	UserId	Itemid	OrderDate
1	5	1	2016-02-14
2	1	1	2016-02-14
3	5	3	2016-02-14
4	1	3	2016-02-14

In OrderHistory table, the UserId data row represents the UserId of a customer from Customers table and the ItemId data row represents the ItemId of an item from the items table. A single customer from Customers table can make multiple orders. Therefore, UserId column of the OrderHistory table can store the same item multiple times. Similarly, a single item from the Items table can be ordered multiple times. Therefore, ItemId column of the OrderHistory table can store the same ItemId multiple times. Take a look at the OrderHistory table, the UserId '5' and 1 have made 2 orders but the user has only one UserId in the Customers table. Similarly, the ItemId '1' and 3 are purchased 2 times but the item has only one ItemId in the items table. This represents Many-To-Many Relationship between the Customers, OrderHistory, and Items tables.

Please use the below SQL Script to create the OnlineShop database, and create the Customers, Items, and OrderHistory tables with the required sample data.

```
CREATE DATABASE OnlineShop;
USE OnlineShop;

-- Create Customers Table
CREATE TABLE Customers(
 UserId INT NOT NULL PRIMARY KEY,
 Name VARCHAR(80) NOT NULL,
 Phone VARCHAR(15),
 Address VARCHAR(150)
);

-- Populate the Customers Table with test data
INSERT INTO Customers VALUES (1, 'Jimmy','4567891230','Mumbai');
INSERT INTO Customers VALUES (2, 'Henry','1234567890','Delhi');
INSERT INTO Customers VALUES (3, 'Ruby','7539514682','BBSR');
INSERT INTO Customers VALUES (4, 'Julia','3571592486','Mumbai');
INSERT INTO Customers VALUES (5, 'Anna','0231456785','Delhi');

-- Create ITEMS Table
CREATE TABLE Items(
 ItemId INT NOT NULL PRIMARY KEY,
 CategoryId INT,
 Name VARCHAR(100) NOT NULL,
 Price FLOAT
);

-- Populate the Items Table with test data
INSERT INTO items VALUES (1, 1, 'Androud Mobile Phone', 250.00);
INSERT INTO items VALUES (2, 1, 'i7 processor, 8GB RAM Laptop',
1000.00);
INSERT INTO items VALUES (3, 2, 'How to train your cat', 25.00);
INSERT INTO items VALUES (4, 2, 'Healthy dog food recipes', 19.00);
INSERT INTO items VALUES (5, 2, 'Learn how to meditate for mental
health', 30.00);
INSERT INTO items VALUES (6, 3, 'Beautiful Black T-Shirts', 99.00);
INSERT INTO items VALUES (7, 3, 'Blue Colored Jeans', 150.00);

-- Create OrderHistory Table
CREATE TABLE OrderHistory (
     OrderHistoryId INT PRIMARY KEY,
    UserId INT NOT NULL,
    Itemid INT NOT NULL,
     OrderDate DATE
);

-- Populate the OrderHistory Table with test data
INSERT INTO OrderHistory VALUES (1, 5, 1, '2016-02-14');
INSERT INTO OrderHistory VALUES (2, 1, 1, '2016-02-14');
INSERT INTO OrderHistory VALUES (3, 5, 3, '2016-02-14');
INSERT INTO OrderHistory VALUES (4, 1, 3, '2016-02-14');
```

In the next article, I am going to discuss Database Normalization in MySQL with Examples. Here, in this article, I try to explain **Database Relationships in MySQL** with Examples and I hope you enjoy this Database Relationships in MySQL article.

← Previous Lesson
SQL Injection in MySQL

Next Lesson →
Database Normalization in MySQL

Leave a Reply

About Privacy Policy Contact ADO.NET Tutorial Angular Tutorials ASP.NET Core Blazor Tuturials ASP.NET Core Tutorials ASP.NET MVC Tutorials ASP.NET Web API Tutorials C Tutorials C#.NET Programs Tutorials C#.NET Tutorials Cloud Computing Tutorials Data Structures and Algorithms Tutorials Design Patterns Tutorials DotNet Interview Questions and Answers Core Java Tutorials Entity Framework Tutorials JavaScript Tutorials LINQ Tutorials Python Tutorials SOLID Principles Tutorials SQL Server Tutorials Trading Tutorials JDBC Tutorials Java Servlets Tutorials Java Struts Tutorials C++ Tutorials JSP Tutorials MySQL Tutorials Oracle Tutorials ASP.NET Core Web API Tutorials HTML Tutorials