My SQL Constraint and Relationship

Objectives

1. Not Null
2. Unique
3. Primary key
4. Foreign key
5. One to many
6. One to one
7. Many to many

1. Not Null Constraint

A column with a NOT NULL constraint, cannot have NULL values.

```
mysql> CREATE TABLE People(Id INTEGER, LastName TEXT NOT NULL,
    ->                     FirstName TEXT NOT NULL, City VARCHAR(55));
Query OK, 0 rows affected (0.07 sec)
```

We create two columns with NOT NULL constraints.

```
mysql> INSERT INTO People VALUES(1, 'Hanks', 'Robert', 'New York');
Query OK, 1 row affected (0.00 sec)

mysql> INSERT INTO People VALUES(1, NULL, 'Marianne', 'Chicago');
ERROR 1048 (23000): Column 'LastName' cannot be null
```

```
The first SELECT statement is executed OK, the second one fails. The SQL error says, the LastName column may not be null.
```

1. Unique Constraint

The UNIQUE constraint ensures that all data are unique in a column.

```
mysql> CREATE TABLE Brands(Id INTEGER, BrandName VARCHAR(30) UNIQUE);
Query OK, 0 rows affected (0.08 sec)
```

Here we create a table Brands. The BrandName column is set to be UNIQUE. There cannot be two brands with the same name.

```
mysql> INSERT INTO Brands VALUES(1, 'Coca Cola');
Query OK, 1 row affected (0.03 sec)

mysql> INSERT INTO Brands VALUES(2, 'Pepsi');
Query OK, 1 row affected (0.00 sec)

mysql> INSERT INTO Brands VALUES(3, 'Pepsi');
ERROR 1062 (23000): Duplicate entry 'Pepsi' for key 'BrandName'
```

3. Primary Key Constraint

The PRIMARY KEY constraint uniquely identifies each record in a database table. It is a special case of unique keys. Primary keys cannot be NULL, unique keys can be. There can be more UNIQUE columns, but only one primary key in a table. Primary keys are important when designing the database tables. Primary keys are unique ids. We use them to refer to table rows. Primary keys become foreign keys in other tables, when creating relations among tables.

```
mysql> DROP TABLE Brands;
mysql> CREATE TABLE Brands(Id INTEGER PRIMARY KEY, BrandName VARCHAR(30) UNIQUE);
```

3. Primary Key Constraint

The Id column of the Brands table becomes a primary key.

```
mysql> DROP TABLE Brands;
mysql> CREATE TABLE Brands(Id INTEGER PRIMARY KEY, BrandName VARCHAR(30) UNIQUE);

mysql> DESCRIBE Brands;
+------------+-------------+------+-----+---------+-------+
| Field      | Type        | Null | Key | Default | Extra |
+------------+-------------+------+-----+---------+-------+
| Id         | int(11)     | NO   | PRI | NULL    |       |
| BranchName | varchar(30) | YES  | UNI | NULL    |       |
+------------+-------------+------+-----+---------+-------+
```

The `DESCRIBE` statement shows information about the columns in a table. We can see that the Id column has a PRIMARY KEY defined and the BrandName has UNIQUE constraint set. The primary key is used to uniquely identify the row in a table, when dealing with a specific table. The unique key enforces that all data in a column are not duplicate.

4. Foreign Key Constraint

A FOREIGN KEY in one table points to a PRIMARY KEY in another table. It is a referential constraint between two tables. The foreign key identifies a column or a set of columns in one (referencing) table that refers to a column or set of columns in another (referenced) table.

We will be demonstrating this constraint on two tables: Authors and Books.

```
mysql> CREATE TABLE Authors(AuthorId INTEGER PRIMARY KEY, Name VARCHAR(70))
    -> type=InnoDB;
```

4. Foreign Key Constraint

Here we create the Authors table. In MySQL, the referencing and the referenced tables must be of InnoDB or BDB storage engines. In the MyISAM storage engines the foreign keys are parsed, but they are not enforced.

```
mysql> CREATE TABLE Books(BookId INTEGER PRIMARY KEY, Title VARCHAR(50),
    -> AuthorId INTEGER, FOREIGN KEY(AuthorId) REFERENCES Authors(AuthorId))
    -> type=InnoDB;
```

We create the Books table. Here we have an AuthorId column name, which acts as a foreign key. It references to the primary key of the Authors table.

5. One to many Relationship

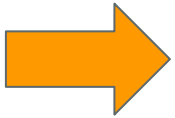
```
mysql> select customerId, name
from customers;
+------------+-------------+
| customerId | name        |
+------------+-------------+
|          1 | Tom Willis  |
|          2 | Terry Neils |
|          3 | Cindy Mason |
|          4 | Paul Novak  |
|          5 | Jack Fonda  |
+------------+-------------+
```

```
mysql> select * from reservations;
+----+------------+------------+--------+
| id | customerId | day        | status |
+----+------------+------------+--------+
|  1 |          4 | 2009-11-22 |      1 |
|  2 |          2 | 2009-11-28 |      1 |
|  3 |          2 | 2009-11-29 |      1 |
|  4 |          4 | 2009-11-29 |      1 |
|  5 |          5 | 2009-12-02 |      1 |
|  6 |          2 | 2009-12-03 |      2 |
|  7 |          3 | 2009-12-04 |      2 |
+----+------------+------------+--------+
```

One customer can has **Many** reservations

5. One to many Relationship

One-to-Many Relationship is implemented using dbo.Reservation(Pk_Id) as the Primary Key and dbo.Customers (Fk_CustomerId) as (Foreign Key). Thus, it will always have only One-to-Many (One Customer-Multiple Reservations) matching rows between the Customer - Reservation table based on the dbo.Customers (Pk_CustomerId) - dbo.Reservations(Fk_Id) relationship.



The One-to-Many relationship is defined as a relationship between two tables where a row from one table can have multiple matching rows in another table. This relationship can be created using Primary key - Foreign key relationship.

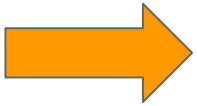
How to create One to Many relationship?

1. Create two Tables (Table A & Table B) with the Primary Key on both the tables.
2. Create a Foreign key in Table B which references the Primary key of Table A.

6. One to one Relationship

```
mysql> select customerId, name
from customers;
+------------+-------------+
| customerId | name        |
+------------+-------------+
|          1 | Tom Willis  |
|          2 | Terry Neils |
|          3 | Cindy Mason |
|          4 | Paul Novak  |
|          5 | Jack Fonda  |
+------------+-------------+
```

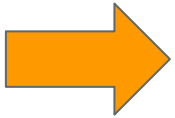
```
mysql> select * from customer_rank;
+----+------------+---------+
| id | customerId | rank_id |
+----+------------+---------+
|  1 |          1 |       3 |
|  2 |          2 |       3 |
|  3 |          3 |       1 |
|  4 |          4 |       2 |
|  5 |          5 |       2 |
+----+------------+---------+
```



One customer can has only **One** rank.

6. One to One Relationship

One-to-One Relationship is implemented using dbo.Customers(Pk_CustomerId) as the Primary key and dbo.Customer_rank(fk_customerId) as (Unique Key Constraint - Foreign Key).



One-to-One (1-1) relationship is defined as the relationship between two tables where both the tables should be associated with each other based on only one matching row. This relationship can be created using Primary key-Unique foreign key constraints.

How to create One to One relationship?

1. Create two Tables (Table A & Table B) with the Primary Key on Both the tables.
2. Create Foreign key in Table B which references the Primary key of Table A.
3. Add a Unique Constraint on the Foreign Key column of Table B.

7. Many to Many Relationship

```
mysql> select * from bills;
+--------+------------+-----------+
| billId | customerId | productId |
+--------+------------+-----------+
|      1 |          2 |         1 |
|      2 |          2 |         5 |
|      3 |          1 |         1 |
|      4 |          1 |         4 |
+--------+------------+-----------+
```

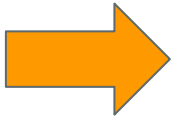
```
mysql> select * from products;
+-----------+-------------+-------+
| productId | productName | price |
+-----------+-------------+-------+
|         1 | Milk        |    50 |
|         2 | Egg         |    30 |
|         3 | Cheese      |    80 |
|         4 | Bread       |    10 |
|         5 | Sugar       |    60 |
+-----------+-------------+-------+
```

```
mysql> select customerId, name
from customers;
+------------+-------------+
| customerId | name        |
+------------+-------------+
|          1 | Tom Willis  |
|          2 | Terry Neils |
|          3 | Cindy Mason |
|          4 | Paul Novak  |
|          5 | Jack Fonda  |
+------------+-------------+
```

Many customers can buy Many products
Or
One customer can buy **Many** products and **One** product can be bought by **Many** Customer

7. Many to Many Relationship

Many-to-many Relationship exists between customers and products: customers can purchase various products, and products can be purchased by many customers.



Many-to-Many relationship is occurs when multiple records in a table are associated with multiple records in another table

=> If you change data in one or both side, you can't know that which record they were referring to.

To avoid this problem, you can break the many-to-many relationship into two one-to-many relationships by using a third table, called a junction table