MYSQL CONSTRAINTS

SQL constraints are used to specify rules for the data in a table. Constraints are used to limit the type of data that can go into a table.

MySQL Primary Key

1. The primary key is a column or a set of columns that uniquely identifies each row in the table. A primary key column must contain unique values.
2. If the primary key consists of multiple columns, the combination of values in these columns must be unique. Additionally, a primary key column cannot contain NULL.
3. A table can have either zero or one primary key, but not more than one.

Defining a single-column primary key

```
CREATE TABLE table_name(
   column1 datatype PRIMARY KEY,
   column2 datatype,
   ...
);
```

Or

```
CREATE TABLE table_name(
   column1 datatype,
   column2 datatype,
   ...,
   PRIMARY KEY(column1)
);
```

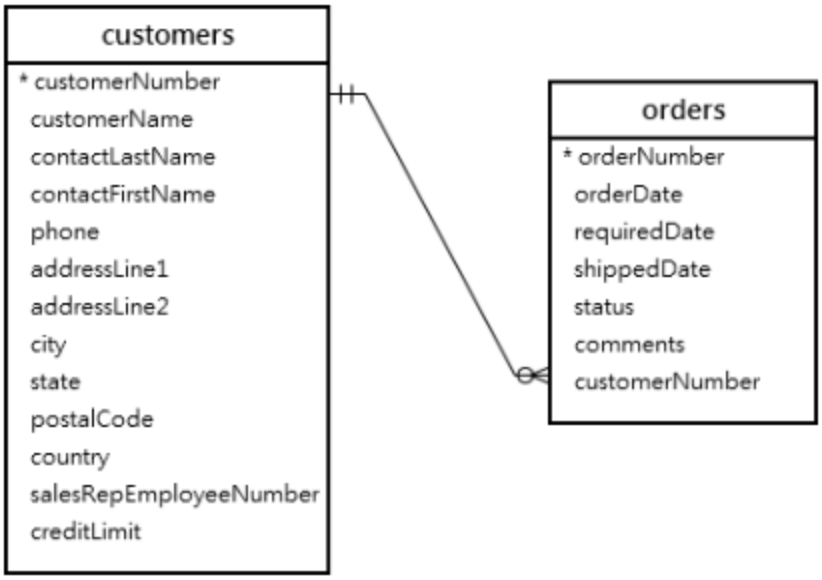
Defining a multi-column primary key

```
CREATE TABLE table_name(
   column1 datatype,
   column2 datatype,
   column3 datatype,
   ...,
   PRIMARY KEY(column1, column2)
);
```

MySQL Foreign Key

A foreign key is a column or group of columns in a table that links to a column or group of columns in another table.

Let’s take a look at the following customers and orders tables



The customerNumber column in the orders table links to the customerNumber primary key column in the customers table.

The customers table is called the parent table or referenced table, and the orders table is known as the child table or referencing table.

Typically, the foreign key columns of the child table often refer to the primary key columns of the parent table.

A table can have more than one foreign key where each foreign key references a primary key of the different parent tables.

MySQL FOREIGN KEY syntax

Here is the basic syntax of defining a foreign key constraint in the CREATE TABLE or ALTER TABLE statement:

```
[CONSTRAINT constraint_name]
FOREIGN KEY [foreign_key_name] (column_name, ...)
REFERENCES parent_table(colunm_name,...)
[ON DELETE reference_option]
[ON UPDATE reference_option]
```

In this syntax:

First, specify the name of the foreign key constraint that you want to create after the CONSTRAINT keyword. If you omit the constraint name, MySQL automatically generates a name for the foreign key constraint.

Second, specify a list of comma-separated foreign key columns after the FOREIGN KEY keywords. The foreign key name is also optional and is generated automatically if you skip it.

Third, specify the parent table followed by a list of comma-separated columns to which the foreign key columns reference.

Finally, specify how the foreign key maintains the referential integrity between the child and parent tables by using the ON DELETE and ON UPDATE clauses. The reference_option determines the action that MySQL will take when values in the parent key columns are deleted (ON DELETE) or updated (ON UPDATE).

MySQL has five reference options: CASCADE, SET NULL, NO ACTION, RESTRICT, and SET DEFAULT.

1. CASCADE: if a row from the parent table is deleted or updated, the values of the matching rows in the child table are automatically deleted or updated.
2. SET NULL: if a row from the parent table is deleted or updated, the values of the foreign key column (or columns) in the child table are set to NULL.
3. RESTRICT: if a row from the parent table has a matching row in the child table, MySQL rejects deleting or updating rows in the parent table.
4. NO ACTION: is the same as RESTRICT.

MySQL fully supports three actions: RESTRICT, CASCADE and SET NULL.

If you don’t specify the ON DELETE and ON UPDATE clause, the default action is RESTRICT.

BONUS:

List All Constraints

```
SELECT
   TABLE_NAME,
   CONSTRAINT_NAME,
   CONSTRAINT_TYPE
FROM
   information_schema.TABLE_CONSTRAINTS
WHERE
   TABLE_SCHEMA = 'your_database_name';
```

List Foreign Key Constraints

```
SELECT
   TABLE_NAME,
   CONSTRAINT_NAME,
   COLUMN_NAME,
   REFERENCED_TABLE_NAME,
   REFERENCED_COLUMN_NAME
FROM
   information_schema.KEY_COLUMN_USAGE
WHERE
   TABLE_SCHEMA = 'your_database_name'
   AND REFERENCED_TABLE_NAME IS NOT NULL;
```

List Constraints for a Specific Table

```
SELECT
   CONSTRAINT_NAME,
   CONSTRAINT_TYPE
FROM
   information_schema.TABLE_CONSTRAINTS
WHERE
   TABLE_SCHEMA = 'your_database_name'
   AND TABLE_NAME = 'your_table_name';
```

MySQL UNIQUE Constraint

Sometimes, you want to ensure values in a column or a group of columns are unique. For example, email addresses of users in the users table, or phone numbers of customers in the customers table should be unique. To enforce this rule, you use a UNIQUE constraint.

A UNIQUE constraint is an integrity constraint that ensures the uniqueness of values in a column or group of columns. A UNIQUE constraint can be either a column constraint or a table constraint.

```
CREATE TABLE table_name(
    ...,
    column1 datatype UNIQUE,
    ...
);
```

To define a UNIQUE constraint for two or more columns, you use the following syntax:

```
CREATE TABLE table_name(
    ...
    column1 datatype,
    column2 datatype,
    ...,
    UNIQUE(column1, column2)
);
```

If you define a UNIQUE constraint without specifying a name, MySQL automatically generates a name for it. To define a UNIQUE constraint with a name, you use this syntax:

```
[CONSTRAINT constraint_name]
UNIQUE(column_list)
```

Key Differences

Nullability:

Primary Key: Cannot contain NULL values.
Unique Constraint: Can contain NULL values.
Number per Table:

Primary Key: Only one primary key per table.
Unique Constraint: Multiple unique constraints can be defined on a table.

Purpose:

Primary Key: Primarily used to uniquely identify records in the table and establish relationships between tables.
Unique Constraint: Used to enforce uniqueness on columns that are not primary identifiers.

MySQL NOT NULL Constraint

A NOT NULL constraint ensures that values stored in a column are not NULL. The syntax for defining a NOT NULL constraint is as follows:

column_name data_type NOT NULL;

A column may have only one NOT NULL constraint, which enforces the rule that the column must not contain any NULL values.

In other words, if you attempt to update or insert a NULL value into a NOT NULL column, MySQL will issue an error.

MySQL DEFAULT

MySQL DEFAULT constraint allows you to specify a default value for a column. Here's the syntax of the DEFAULT constraint:

```
column_name data_type DEFAULT default_value;
```

In this syntax, you specify the DEFAULT keyword followed by the default value for the column. The type of the default value matches the data type of the column.

The default_value must be a literal constant, e.g., a number or a string. It cannot be a function or an expression. However, MySQL allows you to set the current date and time (CURRENT_TIMESTAMP) to the TIMESTAMP and DATETIME columns.

When you define a column without the NOT NULL constraint, the column will implicitly take NULL as the default value.

If a column has a DEFAULT constraint and the INSERT or UPDATE statement doesn't provide the value for that column, MySQL will use the default value specified in the DEFAULT constraint.

Typically, you set the DEFAULT constraints for columns when you create the table. MySQL also allows you to add default constraints to the columns of existing tables. If you don't want to use default values for columns, you can remove the default constraints.

MySQL CHECK Constraint

A CHECK constraint is a very useful tool that can be used in MySQL to prevent invalid data from being inserted into a table. It works by checking the values of certain columns in the table against certain conditions that you specify.

```
CHECK(expression)
```

References

NULL

- **Definition**: `NULL` represents the absence of a value or an unknown value. It is a special marker used in SQL to indicate that a data value does not exist in the database.
- **Behavior**: `NULL` is not equal to anything, not even to another `NULL`. Any comparison with `NULL` using standard comparison operators (`=`, `!=`, `>`, `<`, etc.) results in `NULL`.
- **Usage**: `NULL` is used when a value is missing, unknown, or not applicable.

Empty String (`""`)

- **Definition**: An empty string is a string with zero characters. It is a valid string value, but it contains no characters.
- **Behavior**: An empty string is considered a value and can be compared with other strings. It is equal to another empty string but not to `NULL`.
- **Usage**: An empty string is used to represent a known, but empty, value.