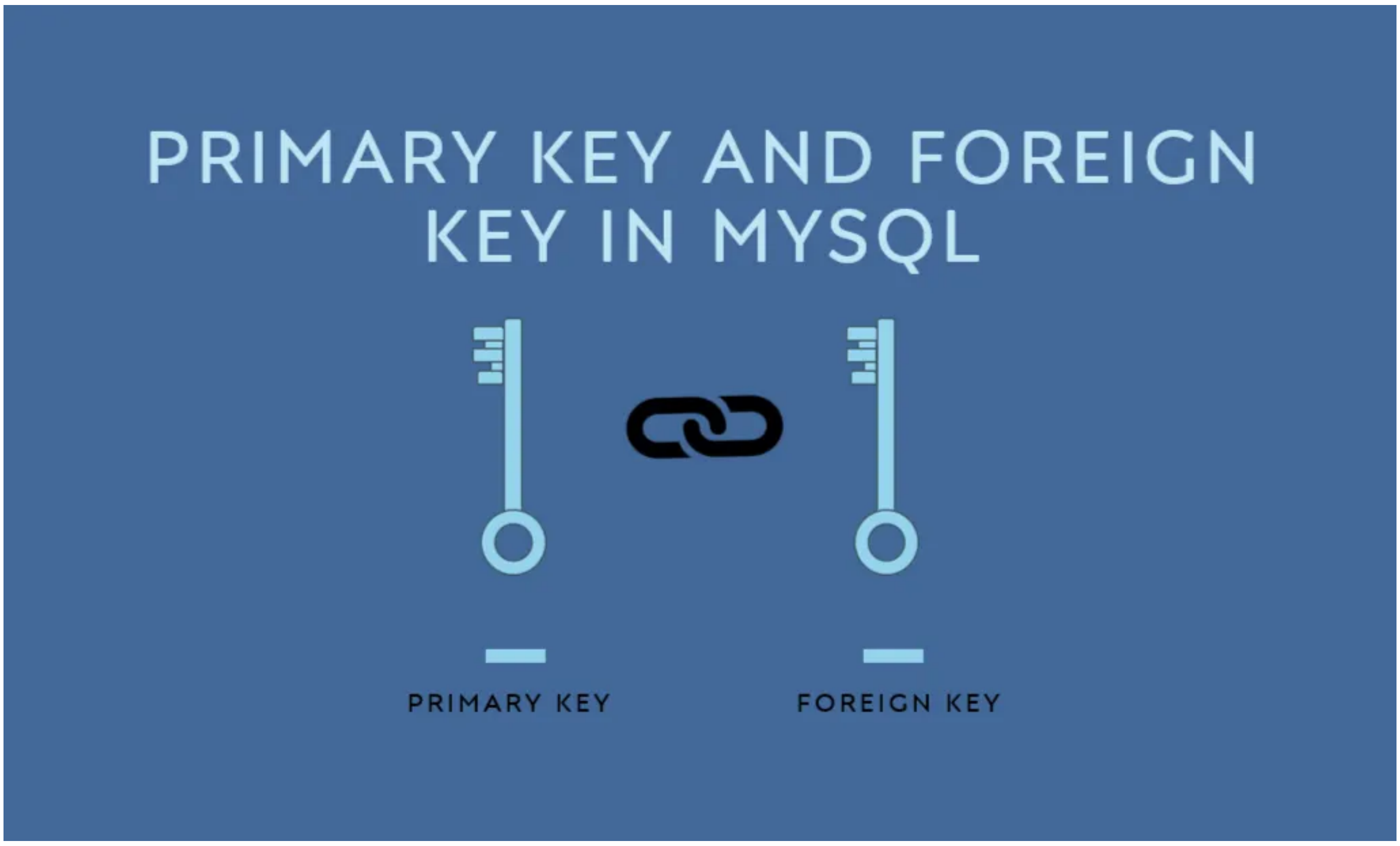
# Building The Basics : Primary key and Foreign key in MySQL

Harshal Yaravalkar Follow 7 min read · Sep 12, 2023

2

Primary key and Foreign key ensure data integrity, maintain relationships between tables, and ultimately, shape the foundation of a well-structured database.



## Abstract

Whether you’re a database enthusiast or a experienced developer looking to solidify your understanding, this blog is your compass through the maze of primary keys and foreign keys in MySQL. Our aim is to provide a clear and concise understanding of these concepts, focusing on their practical significance and implementation.

## Table Of Contents

## PRIMARY KEY

- Primary key is a column or a set of columns that uniquely identifies each row in a table.
- A primary key can be defined using the PRIMARY KEY constraint when

creating or altering a table.

### CREATE table with PRIMARY KEY

Syntax : 1) CREATE TABLE tablename (column1 datatype, column2 datatype, …, primary key (column1));

2) CREATE TABLE tablename (column1 datatype primary key, column2 datatype, …);

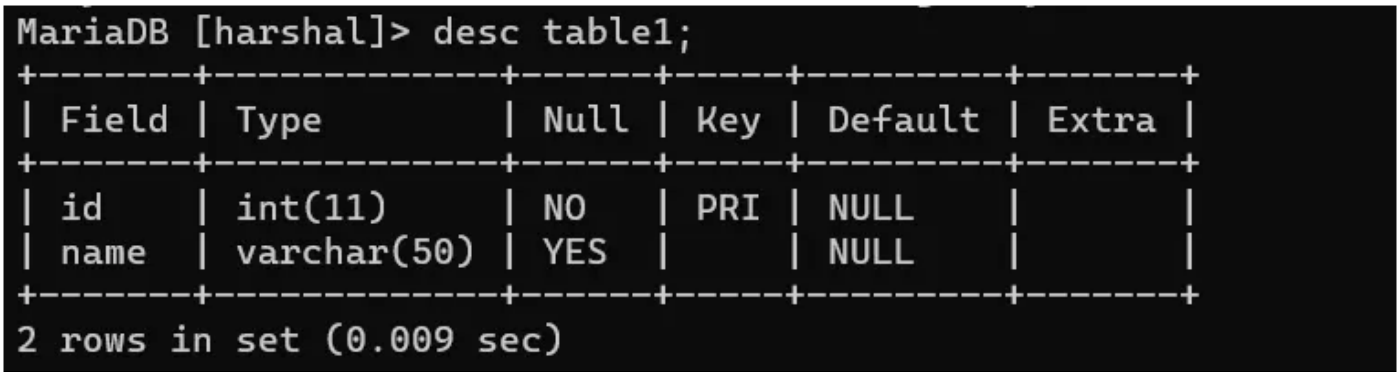
After starting MySQL on Xampp and opening its shell. connect to MySQL using command " MySQL -u root " and create a database of your own name like here I am creating one named "harshal" by entering command " create database harshal;" and then command "use harshal;" for using that database. now, we can create a table.

for example, let's create a table called table1 with two columns: id and name. The id column will be the primary key of this table.

```
CREATE TABLE table1 (

id INT PRIMARY KEY,

name VARCHAR (50)

);
```

let's check if primary key id applied on column by describing table

```
DESC table1;
```



```
MariaDB [harshal]> desc table1;
+-------+-------------+------+-----+---------+-------+
| Field | Type        | Null | Key | Default | Extra |
+-------+-------------+------+-----+---------+-------+
| id    | int(11)     | NO   | PRI | NULL    |       |
| name  | varchar(50) | YES  |     | NULL    |       |
+-------+-------------+------+-----+---------+-------+
2 rows in set (0.009 sec)
```

To drop a primary key, we can use the ALTER TABLE statement with the DROP PRIMARY KEY clause. For example, let's drop the primary key of table1.

```
ALTER TABLE table1 DROP PRIMARY KEY;
```

also, we can add a primary key on desired column of an existing table which doesn't have primary key on it.

### ADD PRIMARY KEY

also, we can add a primary key on desired column of an existing table which doesn't have primary key on it.

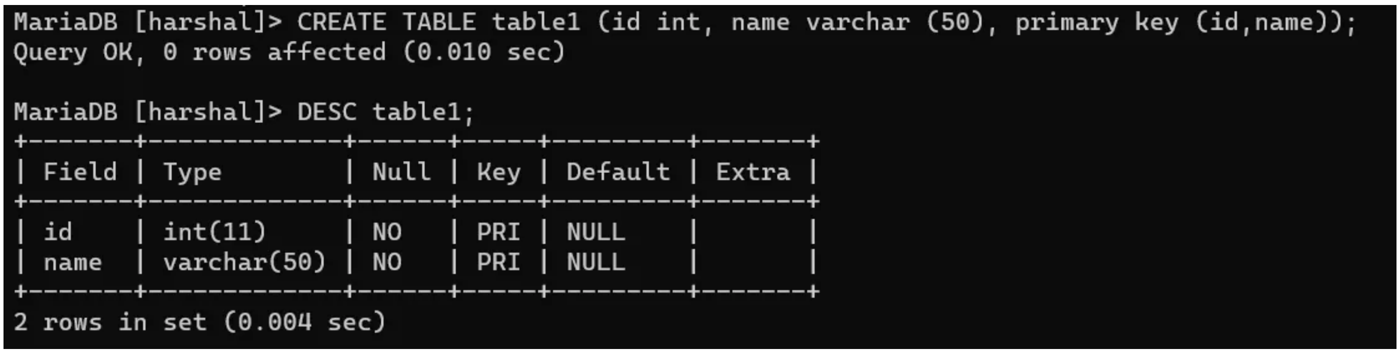
let's add the primary key back on table1.

```
ALTER TABLE table1 ADD PRIMARY KEY (id);
```

### COMPOSITE PRIMARY KEY

In some cases when required we can apply primary key on multiple columns in a composite way

```
CREATE TABLE tablename (
id int,
Name varchar (50),
primary key (id, Name)
);
```



```
MariaDB [harshal]> CREATE TABLE table1 (id int, name varchar (50), primary key (id,name));
Query OK, 0 rows affected (0.010 sec)

MariaDB [harshal]> DESC table1;
+-------+-------------+------+-----+---------+-------+
| Field | Type        | Null | Key | Default | Extra |
+-------+-------------+------+-----+---------+-------+
| id    | int(11)     | NO   | PRI | NULL    |       |
| name  | varchar(50) | NO   | PRI | NULL    |       |
+-------+-------------+------+-----+---------+-------+
2 rows in set (0.004 sec)
```

here both columns possess primary key on them.

now, when foreign key is applied on these composite primary key columns it is only applied on first column having primary key.

## FOREIGN KEY

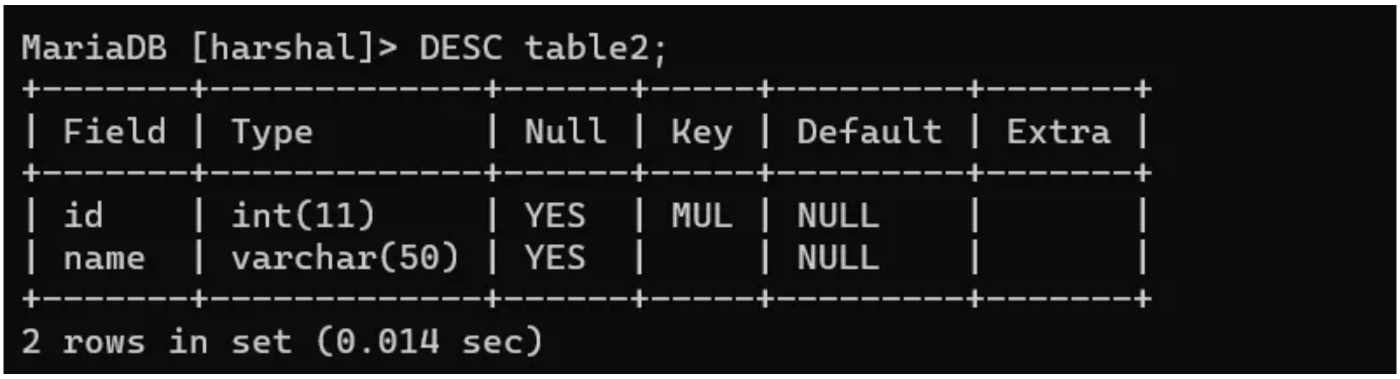
- Foreign key is a column or a set of columns that references another column or a set of columns in another table.
- A foreign key can be defined using the FOREIGN KEY constraint when creating or altering a table.

### CREATE table with FOREIGN KEY

Syntax: CREATE TABLE tablename1 (column_1 datatype, column_2 datatype, foreign key (column_1) references tablename (column1))

Let's create another table called table2 with two columns: id and name. The id column will be the primary key of this table, and the name column will be the foreign key that references the name column of table1.

```
CREATE TABLE table2 (
id INT,
name VARCHAR (50), FOREIGN KEY (id) REFERENCES table1(id)
);
```



```
MariaDB [harshal]> DESC table2;
+-------+-------------+------+-----+---------+-------+
| Field | Type        | Null | Key | Default | Extra |
+-------+-------------+------+-----+---------+-------+
| id    | int(11)     | YES  | MUL | NULL    |       |
| name  | varchar(50) | YES  |     | NULL    |       |
+-------+-------------+------+-----+---------+-------+
2 rows in set (0.014 sec)
```

### DROP FOREIGN KEY

To drop a foreign key, we need to know its name first. We can use the SHOW CREATE TABLE statement to see the name of the foreign key constraint. For example, let's see the name of the foreign key of table2.

```
SHOW CREATE TABLE table2;
```

The output will show something like this:

CREATE TABLE `table2` (

`id` int (11) DEFAULT NULL,

`name` varchar (50) DEFAULT NULL,

KEY `id` (`id`),

CONSTRAINT `table2_ibfk_1` FOREIGN KEY (`id`) REFERENCES `table1` (`id`)

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_general_ci

The name of the foreign key constraint is `table2_ibfk_1`. To drop it, we can use the ALTER TABLE statement with the DROP FOREIGN KEY clause. For example, let's drop the foreign key of table2.

```
ALTER TABLE table2 DROP FOREIGN KEY table2_ibfk_1;
```

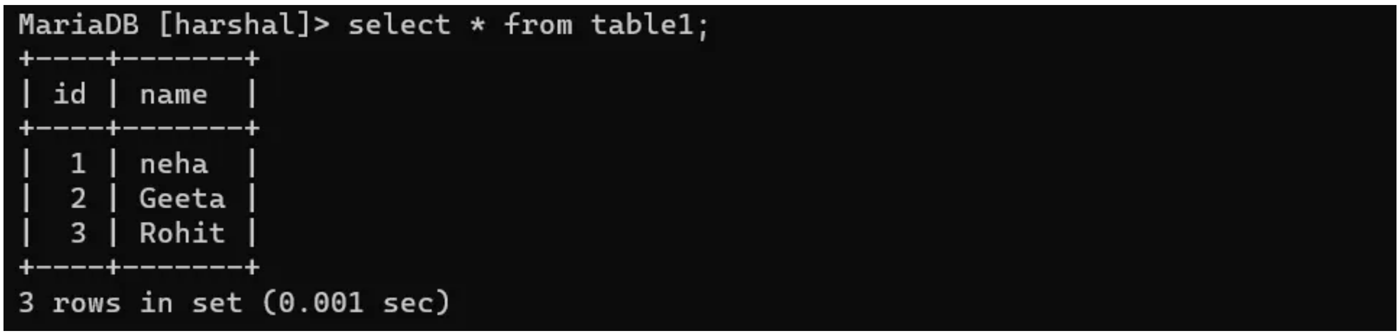
### Applying ON DELETE CASCADE and ON DELETE NULL on FOREIGN KEY

One of the benefits of using foreign keys is that they can enforce referential integrity between tables. This means that if a row in one table is deleted or updated, the corresponding rows in another table will be affected accordingly. There are different options to specify how foreign keys should behave on delete or update actions. These options are called referential actions and they can be defined using the ON DELETE or ON UPDATE clauses when creating or altering a foreign key constraint. The most common referential actions are CASCADE and SET NULL.

CASCADE means that if a row in the referenced table is deleted or updated, the matching rows in the referencing table will be deleted or updated as well. For example, let's create a third table called table3 with two columns: id and name. The id column will be the foreign key that references the id column of table1 with ON DELETE CASCADE option.

```
CREATE TABLE table3 (
id INT,
name VARCHAR(50), FOREIGN KEY (id) REFERENCES table1(id) ON DELETE CASCADE
);
```

Here, we will need to insert some values of id and name in table1 as well as table3

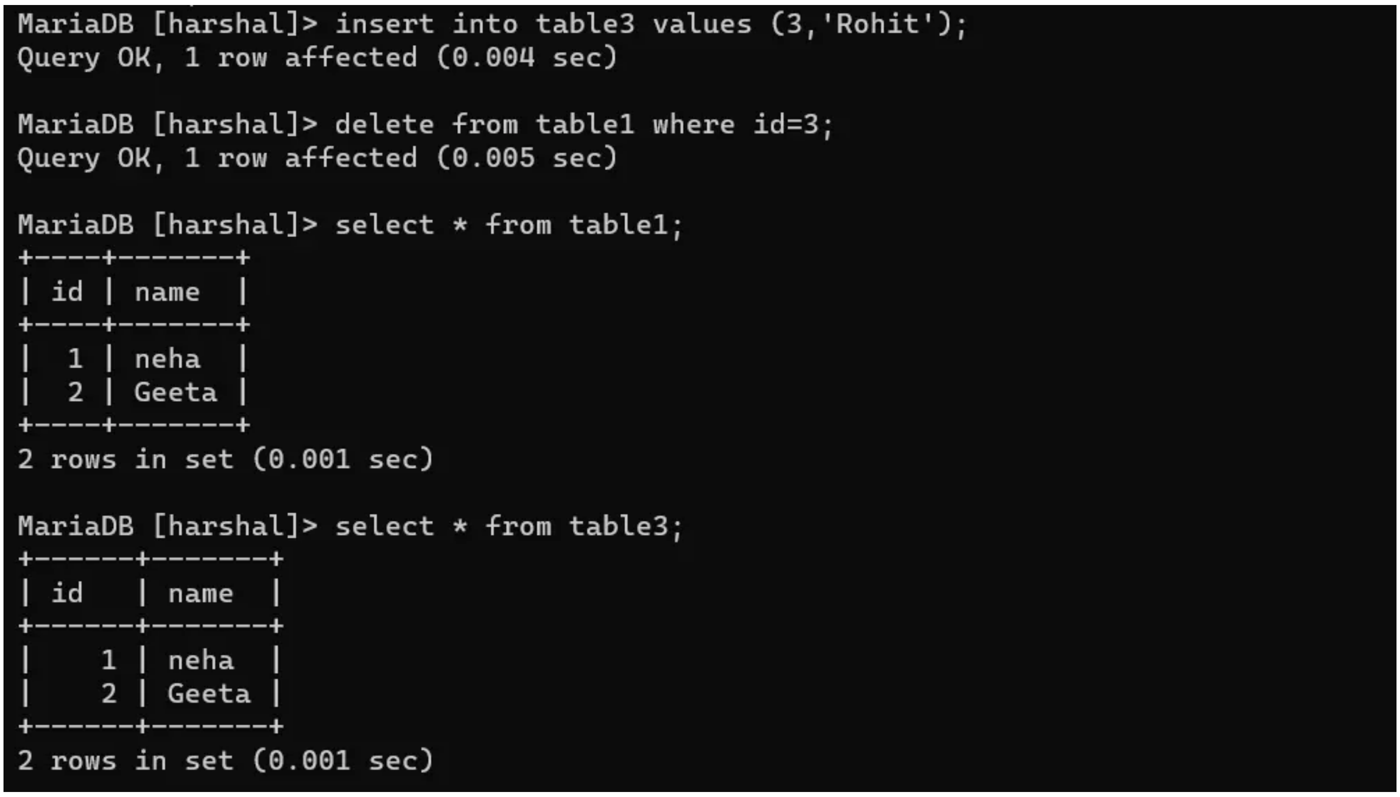


```
MariaDB [harshal]> select * from table1;
+----+-------+
| id | name  |
+----+-------+
|  1 | neha  |
|  2 | Geeta |
|  3 | Rohit |
+----+-------+
3 rows in set (0.001 sec)
```

have inserted values in table1 as well as table3

Now, if we delete a row from table1, the matching row in table3 will be deleted as well. For example, let’s delete the row with id = 3 from table1.

```
DELETE FROM table1 WHERE id = 3;
```



```
MariaDB [harshal]> insert into table3 values (3,'Rohit');
Query OK, 1 row affected (0.004 sec)

MariaDB [harshal]> delete from table1 where id=3;
Query OK, 1 row affected (0.005 sec)

MariaDB [harshal]> select * from table1;
+----+-------+
| id | name  |
+----+-------+
|  1 | neha  |
|  2 | Geeta |
+----+-------+
2 rows in set (0.001 sec)

MariaDB [harshal]> select * from table3;
+------+-------+
| id   | name  |
+------+-------+
|    1 | neha  |
|    2 | Geeta |
+------+-------+
2 rows in set (0.001 sec)
```

This will also delete the row with id = 3 from table3.

SET NULL means that if a row in the referenced table is deleted or updated, the matching rows in the referencing table will have their foreign key columns set to NULL. For example, let’s alter the foreign key of table3 to have ON DELETE SET NULL option instead of ON DELETE CASCADE.

```
ALTER TABLE table3 DROP FOREIGN KEY table3_ibfk_1;
```

I'll explain ON DELETE NULL concept combined with how to use ADD FOREIGN KEY clause by using ALTER TABLE statement in this next point.

### ADD FOREIGN KEY

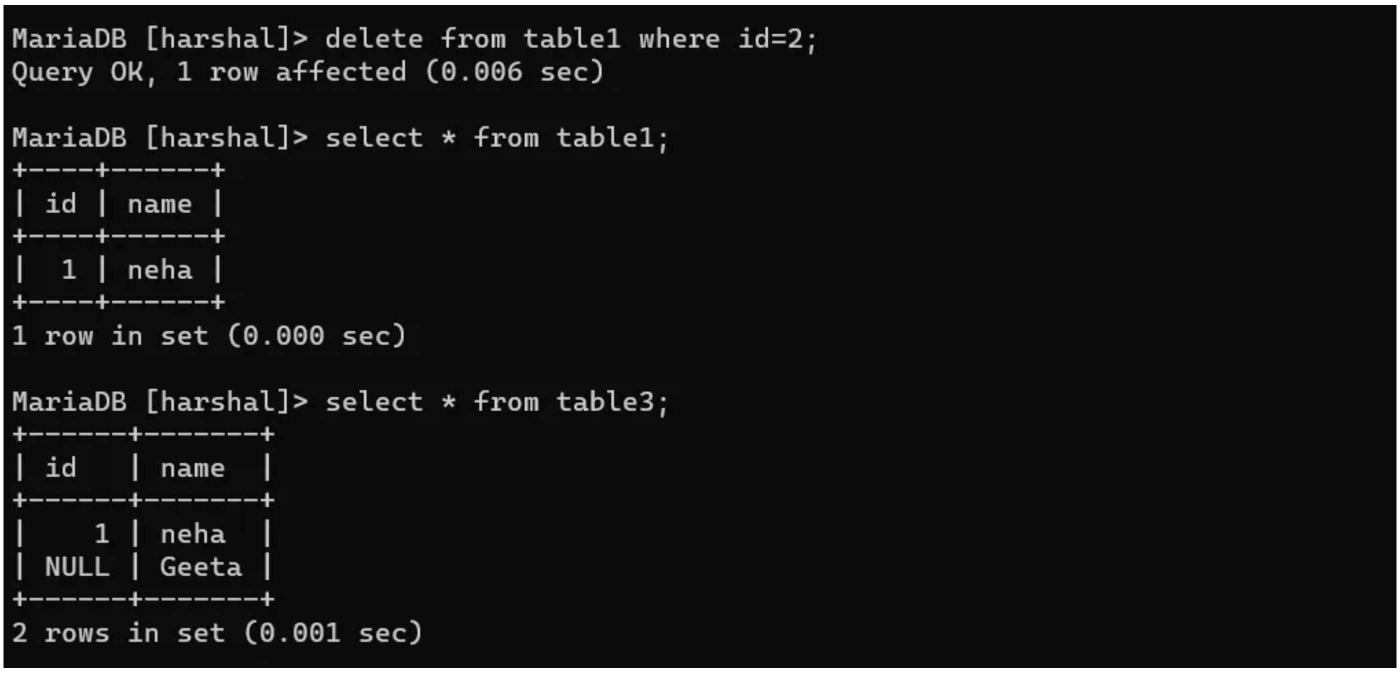
```
ALTER TABLE table3 ADD CONSTRAINT table3_ibfk_1 FOREIGN KEY (id) REFERENCES ta
```

P.S. — in this above command while adding the constraint by ALTER or CREATE we can set the constraint name of our choice . here it is set as "table3_ibfk_1" but it can be even my own name like "harshal" or something technical like "table3fk_to_table1pk" , short or long. for CREATE command it will be

```
CREATE table table4 ( id INT, name VARCHAR(50), CONSTRAINT harshal FOREIGN KEY
```

Now, if we delete a row from table1, the matching row in table3 will have its id column set to NULL. For example, let's delete the row with id = 2 from table1.

```
DELETE FROM table1 WHERE id = 2;
```


```
MariaDB [harshal]> delete from table1 where id=2;
Query OK, 1 row affected (0.006 sec)

MariaDB [harshal]> select * from table1;
+----+------+
| id | name |
+----+------+
|  1 | neha |
+----+------+
1 row in set (0.000 sec)

MariaDB [harshal]> select * from table3;
+------+-------+
| id   | name  |
+------+-------+
|    1 | neha  |
| NULL | Geeta |
+------+-------+
2 rows in set (0.001 sec)
```

This sets the id column of the row with id = 2 in table3 to NULL.

**Joining three or more tables with PRIMARY KEY and FOREIGN KEY**

Another benefit of using foreign keys is that they can enable us to join multiple tables based on their relationships.

*I.* The first example is to apply foreign key on id column of Ctable taking reference from id column of Atable and id column of Btable which both have primary key applied on respective tables. This means that each row in Ctable must have an id that matches a row in either Atable or Btable.

For example, let's create Ctable with this foreign key.

let's do this in steps -

1) CREATE TABLE Atable ( id INT PRIMARY KEY, name VARCHAR(50));

2) CREATE TABLE Btable ( id INT PRIMARY KEY, name VARCHAR(50));

3) CREATE TABLE Ctable ( id INT, name VARCHAR(50), FOREIGN KEY (id) REFERENCES Atable(id), FOREIGN KEY (id) REFERENCES Btable(id));

here, id column of Ctable accepts values that are in id column of Atable and id column of Btable. Even if there are some uncommon id records in Atable and Btable it accepts them.

***II.*** The second example is to apply foreign keys on id column and name column of Ctable taking reference from id column of Atable and name column of Btable which both have primary key applied on them in their respective tables. This means that each row in Ctable must have a unique combination of id and name that matches a row in Atable and a row in Btable. For example, let's create Ctable with these foreign keys.

as earlier we will execute in steps -

1) CREATE TABLE Atable ( id INT PRIMARY KEY, name VARCHAR(50));

2) CREATE TABLE Btable ( id INT, name VARCHAR(50) PRIMARY KEY);

3) CREATE TABLE Ctable ( id INT, name VARCHAR(50), FOREIGN KEY (id) REFERENCES Atable(id), FOREIGN KEY (name) REFERENCES Btable(name));

here, id and name column of Ctable will only accept the records which are present in id column of Atable and name column of Btable.

I hope this helps you get started!

I'll be back next time with some other concept

Till then, ***BUILD YOUR BASICS !!!***

MySQL · Primary Key Sql · Data Science · Database · Sql

Written by Harshal Yaravalkar

0 followers · 1 following

Follow

studying Data science, AI, ML. Avid learner, Graduated in Physics