SQL Joins: ON DELETE CASCADE, ON UPDATE CASCADE

```
CREATE TABLE Payment
(
payment_id int(10) PRIMARY KEY NOT NULL,
emp_id int(10) NOT NULL,
amount float NOT NULL,
payment_date date NOT NULL,
FOREIGN KEY (emp_id)
REFERENCES Employee (emp_id) ON DELETE CASCADE ON UPDATE CASCADE
-- ask this to your manager before implementing
);
```

MYSQL ON DELETE CASCADE

`ON DELETE CASCADE` clause in MySQL is used to automatically remove the matching records from the child table when we delete the rows from the parent table.

Suppose we have created two tables with a FOREIGN KEY in a foreign key relationship, making both tables a parent and child. Next, we define an `ON DELETE CASCADE` clause for one `FOREIGN KEY` that must be set for the other to succeed in the cascading operations.

If the `ON DELETE CASCADE` is defined for one `FOREIGN KEY` clause only, then cascading operations will throw an error.

EXAMPLE

- TABLE EMPLOYEE

```
CREATE TABLE Employee (
 emp_id int(10) NOT NULL,
 name varchar(40) NOT NULL,
 birthdate date NOT NULL,
 gender varchar(10) NOT NULL,
 hire_date date NOT NULL,
 PRIMARY KEY (emp_id)
 );
```

```
mysql> SELECT * FROM Employee;
+--------+--------+------------+--------+------------+
| emp_id | name   | birthdate  | gender | hire_date  |
+--------+--------+------------+--------+------------+
|    101 | Bryan  | 1988-08-12 | M      | 2015-08-26 |
|    102 | Joseph | 1978-05-12 | M      | 2014-10-21 |
|    103 | Mike   | 1984-10-13 | M      | 2017-10-28 |
|    104 | Daren  | 1979-04-11 | M      | 2006-11-01 |
|    105 | Marie  | 1990-02-11 | F      | 2018-10-12 |
+--------+--------+------------+--------+------------+
```

- TABLE PAYMENT

```
CREATE TABLE Payment (
 payment_id int(10) PRIMARY KEY NOT NULL,
 emp_id int(10) NOT NULL,
 amount float NOT NULL,
 payment_date date NOT NULL,
 FOREIGN KEY (emp_id) REFERENCES Employee (emp_id) ON DELETE CASCADE
 );
```

```
mysql> SELECT * FROM Payment;
+------------+--------+--------+--------------+
| payment_id | emp_id | amount | payment_date |
+------------+--------+--------+--------------+
|        301 |    101 |   1200 | 2015-09-15   |
|        302 |    101 |   1200 | 2015-09-30   |
|        303 |    101 |   1500 | 2015-10-15   |
|        304 |    101 |   1500 | 2015-10-30   |
|        305 |    102 |   1800 | 2015-09-15   |
|        306 |    102 |   1800 | 2015-09-30   |
+------------+--------+--------+--------------+
```

- Let us delete data from the parent table Employee. To do this, execute the following statement:

```
DELETE FROM Employee WHERE emp_id = 102;
```

The above statement will delete the employee records whose emp_id = 102 and referencing data into the child table.

- NOW USE SELECT STATEMENT TO VERIFY

```
SELECT * FROM EMPLOYEE;

SELECT * FROM PAYMENT;
```

```
mysql> SELECT * FROM Employee;
+--------+-------+------------+--------+------------+
| emp_id | name  | birthdate  | gender | hire_date  |
+--------+-------+------------+--------+------------+
|    101 | Bryan | 1988-08-12 | M      | 2015-08-26 |
|    103 | Mike  | 1984-10-13 | M      | 2017-10-28 |
|    104 | Daren | 1979-04-11 | M      | 2006-11-01 |
|    105 | Marie | 1990-02-11 | F      | 2018-10-12 |
+--------+-------+------------+--------+------------+
4 rows in set (0.00 sec)

mysql> SELECT * FROM Payment;
+------------+--------+--------+--------------+
| payment_id | emp_id | amount | payment_date |
+------------+--------+--------+--------------+
|        301 |    101 |   1200 | 2015-09-15   |
|        302 |    101 |   1200 | 2015-09-30   |
|        303 |    101 |   1500 | 2015-10-15   |
|        304 |    101 |   1500 | 2015-10-30   |
+------------+--------+--------+--------------+
```

In the above output, we can see that all the rows referencing to emp_id = 102 were automatically deleted from both tables.

As you will work will millions of data so there will be lot of table so to see from which table your record are deleted.

How to find the affected table by ON DELETE CASCADE action?

- Sometimes, before deleting records from the table, we want to know the affected table by the ON DELETE CASCADE referential action. We can find this information by querying from the referential_constraints in the information_schema database as follows:

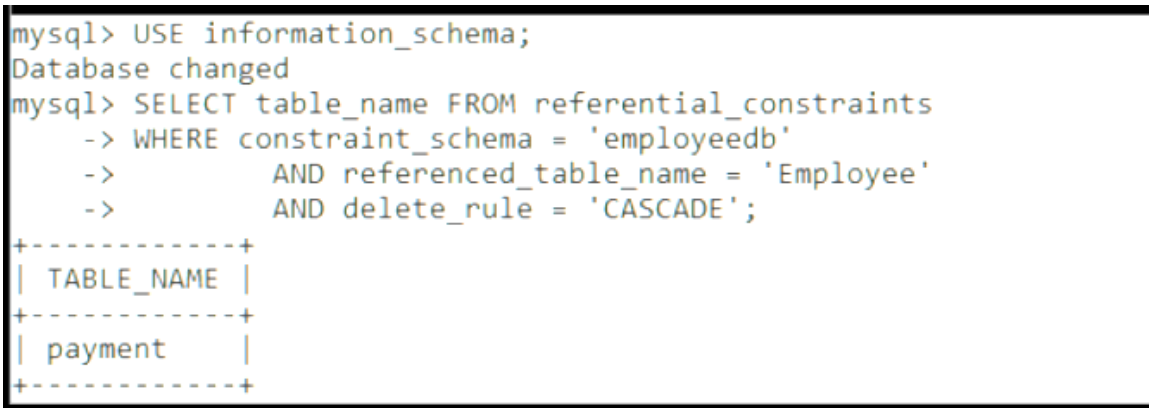
```
USE information_schema;

 SELECT table_name FROM referential_constraints
 WHERE constraint_schema = 'database_name'
   AND referenced_table_name = 'parent_table'
   AND delete_rule = 'CASCADE'
```

  - EXAMPLE:

```
USE information_schema;

 SELECT table_name FROM referential_constraints
 WHERE constraint_schema = 'employeedb'
   AND referenced_table_name = 'Employee'
   AND delete_rule = 'CASCADE';
```



ON UPDATE CASCADE

`ON UPDATE CASCADE` is a referential integrity constraint that tells the database to automatically update the corresponding rows in a child table when a row in the parent table is updated. This ensures that the data in the two tables remains consistent.

Here is an example of how to use the `ON UPDATE CASCADE` clause:

```
CREATE TABLE departments (
  id INT NOT NULL PRIMARY KEY,
  name VARCHAR(255) NOT NULL
);

CREATE TABLE employees (
  id INT NOT NULL PRIMARY KEY,
  name VARCHAR(255) NOT NULL,
  department_id INT NOT NULL REFERENCES departments (id) ON UPDATE CASCADE
);
```

In this example, the `ON UPDATE CASCADE` clause on the `department_id` column in the `employees` table tells the database that when the `department_id` column in the `departments` table is updated, the corresponding rows in the `employees` table will also be updated to reflect the new department ID.

For example, if we update the `department_id` column in the `departments` table to 10, all rows in the `employees` table where the `department_id` column is currently 10 will also be updated to 10.

The `ON UPDATE CASCADE` clause is a powerful tool that can be used to ensure that the data in related tables is always consistent.

Here is an explanation of the `ON UPDATE CASCADE` clause:

- `ON UPDATE` specifies that the constraint is triggered when a row in the parent table is updated.
- `CASCADE` specifies that the corresponding rows in the child table are updated to reflect the changes in the parent table.