REPLACE

This lesson discusses the REPLACE clause.

REPLACE

REPLACE is much like the **INSERT** statement with one key difference: we can't insert a row if a table already contains a row with the same primary key. However, REPLACE allows us the convenience of adding a row with the same primary key as an existing row in the table. Under the hood, REPLACE deletes the row and then adds the new row thereby maintaining the primary key constraint at all times. Sure, we can also use the **UPDATE** clause to achieve the same effect. However, REPLACE can be useful in automated scripts where it is not known ahead of time if a particular table already contains a particular primary key. If it doesn't, the replacement behaves like an insertion, otherwise, it deletes and writes in the new row with the same primary key.

> REPLACE INTO **table (col1, col2, ... coln)**
>
> VALUES **(val1, val2, ... valn)**
>
> WHERE <**condition**>

> Connect to the terminal below by clicking in the widget. Once connected, the command line prompt will show up. Enter or copy and paste the command **./DataJek/Lessons/38lesson.sh** and wait for the MySQL prompt to start-up.

```
-- The lesson queries are reproduced below for convenient copy/paste into the terminal.

-- Query 1
REPLACE INTO
Actors (Id, FirstName, SecondName,DoB, Gender, MaritalStatus, NetworthInMillions)
VALUES (3, "George", "Clooney", "1961-05-06","Male", "Married", 500.00);

-- Query 2
REPLACE INTO
Actors (Id)
VALUES (3);

-- Query 3
REPLACE INTO  Actors
SET id = (SELECT Id
          FROM Actors
          WHERE FirstName="Brad");
```

Terminal

1. We can use all the variations of the **INSERT** clause with **REPLACE** too. Let's start with a simple example, where we want to replace the actor with the ID equal to 3 in the **Actors** table.

```
REPLACE INTO
Actors (Id, FirstName, SecondName,DoB, Gender, MaritalStatus, Net
worthInMillions)
VALUES (3, "George", "Clooney", "1961-05-06","Male", "Married", 5
00.00);
```

```
mysql> REPLACE INTO
    -> Actors (Id, FirstName, SecondName,DoB, Gender, MaritalStatus, NetworthInMillions)
    -> VALUES (3, "George", "Clooney", "1961-05-06","Male", "Married", 500.00);
Query OK, 2 rows affected (0.00 sec)

mysql> SELECT * FROM Actors;
+----+-----------+------------+------------+--------+---------------+--------------------+
| Id | FirstName | SecondName | DoB        | Gender | MaritalStatus | NetWorthInMillions |
+----+-----------+------------+------------+--------+---------------+--------------------+
|  1 | Brad      | Pitt       | 1963-12-18 | Male   | Single        |                240 |
|  2 | Jennifer  | Aniston    | 1969-11-02 | Female | Single        |                240 |
|  3 | George    | Clooney    | 1961-05-06 | Male   | Married       |                500 |
|  4 | Johnny    | Depp       | 1963-06-09 | Male   | Single        |                200 |
|  5 | Natalie   | Portman    | 1981-06-09 | Male   | Married       |                 60 |
|  6 | Tom       | Cruise     | 1962-07-03 | Male   | Divorced      |                570 |
|  7 | Kylie     | Jenner     | 1997-08-10 | Female | Married       |               1000 |
|  8 | Kim       | Kardashian | 1980-10-21 | Female | Married       |                370 |
|  9 | Amitabh   | Bachchan   | 1942-10-11 | Male   | Married       |                400 |
| 10 | Shahrukh  | Khan       | 1965-11-02 | Male   | Married       |                600 |
| 11 | priyanka  | Chopra     | 1982-07-18 | Female | Married       |                 28 |
+----+-----------+------------+------------+--------+---------------+--------------------+
11 rows in set (0.00 sec)
```

You can observe that the output of the replace query says 2 rows affected, which implies one row was deleted and a second was inserted.

2. Now we'll repeat the previous query but only provide the value for the primary key column and observe the outcome.

```
REPLACE INTO
Actors (Id)
VALUES (3);
```

```
mysql> REPLACE INTO
    -> Actors (Id)
    -> VALUES (3);
Query OK, 2 rows affected (0.00 sec)

mysql> SELECT * FROM Actors;
+----+-----------+------------+------------+--------+---------------+--------------------+
| Id | FirstName | SecondName | DoB        | Gender | MaritalStatus | NetWorthInMillions |
+----+-----------+------------+------------+--------+---------------+--------------------+
|  1 | Brad      | Pitt       | 1963-12-18 | Male   | Single        |                240 |
|  2 | Jennifer  | Aniston    | 1969-11-02 | Female | Single        |                240 |
|  3 | NULL      | NULL       | NULL       | NULL   | NULL          |               NULL |
|  4 | Johnny    | Depp       | 1963-06-09 | Male   | Single        |                200 |
|  5 | Natalie   | Portman    | 1981-06-09 | Male   | Married       |                 60 |
|  6 | Tom       | Cruise     | 1962-07-03 | Male   | Divorced      |                570 |
|  7 | Kylie     | Jenner     | 1997-08-10 | Female | Married       |               1000 |
|  8 | Kim       | Kardashian | 1980-10-21 | Female | Married       |                370 |
|  9 | Amitabh   | Bachchan   | 1942-10-11 | Male   | Married       |                400 |
| 10 | Shahrukh  | Khan       | 1965-11-02 | Male   | Married       |                600 |
| 11 | priyanka  | Chopra     | 1982-07-18 | Female | Married       |                 28 |
+----+-----------+------------+------------+--------+---------------+--------------------+
11 rows in set (0.00 sec)
```

As you can see, the rest of the columns not specified in the query end up taking the default value which is NULL.

3. If a table doesn't have a primary key defined, REPLACE behaves exactly like an INSERT. Without a primary key REPLACE can't uniquely identify a row to replace.

4. Remember that when inserting the duplicate row using the **INSERT IGNORE** clause, the duplicate row is ignored and not added to the table whereas when using **REPLACE** the existing row is deleted and the duplicate row is added to the table.

5. Similar to multi-delete and update, we can't replace into a table that is also being read from a subquery. For instance, the following query, which replaces the ID of a row with itself gives an error:

```
REPLACE INTO  Actors
```

```
SET id = (SELECT Id


          FROM Actors
          WHERE FirstName="Brad");
```

```
mysql> REPLACE INTO  Actors
    -> SET id = (SELECT Id
    ->           FROM Actors
    ->           WHERE FirstName="Brad");
ERROR 1093 (HY000): You can't specify target table 'Actors' for update in FROM clause
```