MySQLTUTORIAL

MySQL CHECK Constraint Emulation

Summary: in this tutorial, you will learn how to use emulate `CHECK` constraints in MySQL using triggers or views.

> MySQL 8.0.16 fully implemented the SQL `CHECK` constraint. If you use MySQL 8.0.16 or later, check it out the `CHECK` constraint (https://www.mysqltutorial.org/mysql-check-constraint/) tutorial.

Emulating CHECK constraints using triggers

To emulate `CHECK` constraints in MySQL, you can use two triggers (https://www.mysqltutorial.org/mysql-triggers.aspx) : `BEFORE INSERT` and `BEFORE UPDATE` .

First, create a new table (https://www.mysqltutorial.org/mysql-create-table/) named `parts` for the demonstration:

```
CREATE TABLE IF NOT EXISTS parts (
    part_no VARCHAR(18) PRIMARY KEY,
    description VARCHAR(40),
    cost DECIMAL(10 , 2 ) NOT NULL,
    price DECIMAL(10,2) NOT NULL
);
```

Next, create a stored procedure (https://www.mysqltutorial.org/mysql-stored-procedure-tutorial.aspx) to check the values in the `cost` and `price` columns.

```
DELIMITER $

CREATE PROCEDURE `check_parts`(IN cost DECIMAL(10,2), IN price DECIMAL(10,2))
BEGIN
    IF cost < 0 THEN
        SIGNAL SQLSTATE '45000'
           SET MESSAGE_TEXT = 'check constraint on parts.cost failed';
```

```
    END IF;

    IF price < 0 THEN
        SIGNAL SQLSTATE '45001'
           SET MESSAGE_TEXT = 'check constraint on parts.price failed';
    END IF;

    IF price < cost THEN
        SIGNAL SQLSTATE '45002'
           SET MESSAGE_TEXT = 'check constraint on parts.price & parts.cost failed';
    END IF;
END$
DELIMITER ;
```

Then, create `BEFORE INSERT` and `BEFORE UPDATE` triggers. Inside the triggers, call the `check_parts()` stored procedure.

```
-- before insert
DELIMITER $
CREATE TRIGGER `parts_before_insert` BEFORE INSERT ON `parts`
FOR EACH ROW
BEGIN
    CALL check_parts(new.cost,new.price);
END$
DELIMITER ;
-- before update
DELIMITER $
CREATE TRIGGER `parts_before_update` BEFORE UPDATE ON `parts`
FOR EACH ROW
BEGIN
    CALL check_parts(new.cost,new.price);
END$
DELIMITER ;
```

After that, insert (https://www.mysqltutorial.org/mysql-insert-statement.aspx) a new row that satisfies all the following conditions:

- cost > 0
- And price > 0
- And price >= cost

```
INSERT INTO parts(part_no, description,cost,price)
VALUES('A-001','Cooler',100,120);
```

```
1 row(s) affected
```

The `INSERT` statement invokes the `BEFORE INSERT` trigger and accepts the values.

The following `INSERT` statement fails because it violates the condition: cost > 0.

```
INSERT INTO parts(part_no, description,cost,price)
VALUES('A-002','Heater',-100,120);
```

```
Error Code: 1644. check constraint on parts.cost failed
```

The following `INSERT` statement fails because it violates the condition: price > 0.

```
INSERT INTO parts(part_no, description,cost,price)
VALUES('A-002','Heater',100,-120);
```

```
Error Code: 1644. check constraint on parts.price failed
```

The following `INSERT` statement fails because it violates the condition: price > cost.

```
INSERT INTO parts(part_no, description,cost,price)
VALUES('A-003','wiper',120,100);
```

Let's see what we are having now in the `parts` table.

```
SELECT * FROM parts;
```

The following statement attempt to update the cost to make it lower than the price:

```
UPDATE parts
SET price = 10
WHERE part_no = 'A-001';
```

```
Error Code: 1644. check constraint on parts.price & parts.cost failed
```

The statement was rejected.

So by using two triggers: `BEFORE INSERT` and `BEFORE UPDATE`, you are able to emulate `CHECK` constraints in MySQL.

Emulate CHECK constraints using views

The idea is to create a view `WITH CHECK OPTION` based on the underlying table. In the `SELECT` (https://www.mysqltutorial.org/mysql-select-statement-query-data.aspx) statement of the view definition, we select only valid rows that satisfy the `CHECK` conditions. Any insert or update against the view will be rejected if it causes the new row to not appear in the view.

First, drop the `parts` table to remove all the associated triggers and create a new table (https://www.mysqltutorial.org/mysql-create-table/) like the `parts` table but have a different name `parts_data`:

```
DROP TABLE IF EXISTS parts;

CREATE TABLE IF NOT EXISTS parts_data (
    part_no VARCHAR(18) PRIMARY KEY,
    description VARCHAR(40),
    cost DECIMAL(10,2) NOT NULL,
```

```
    price DECIMAL(10,2) NOT NULL
);
```

Next, create a view (https://www.mysqltutorial.org/create-sql-views-mysql.aspx) named `parts` based on the `parts_data` table. By doing this, we can keep the code of the applications that use the `parts` table remains intact. In addition, all the privileges to the old `parts` table remains unchanged.

```
CREATE VIEW parts AS
    SELECT
        part_no, description, cost, price
    FROM
        parts_data
    WHERE
        cost > 0 AND price > 0 AND price >= cost
WITH CHECK OPTION;
```

Then, insert a new row (https://www.mysqltutorial.org/mysql-insert-statement.aspx) into the `parts_data` table through the `parts` view:

```
INSERT INTO parts(part_no, description,cost,price)
VALUES('A-001','Cooler',100,120);
```

It is accepted because the new row is valid which appears in the view.

After that, attempt to insert a new row that would not appear in the view.

```
INSERT INTO parts_checked(part_no, description,cost,price)
VALUES('A-002','Heater',-100,120);
```

```
Error Code: 1369. CHECK OPTION failed 'classicmodels.parts_checked'
```

In this tutorial, you have learned how to use triggers or views to emulate the `CHECK` constraints in MySQL.