MySQL CHECK Constraint

Summary: in this tutorial, you will learn how to use MySQL `CHECK` constraint to ensure that values stored in a column or group of columns satisfy a Boolean expression.

> MySQL 8.0.16 implemented the SQL check constraint. If you use MySQL with the earlier versions, you can emulate a `CHECK` constraint using a view `WITH CHECK OPTION` or a trigger.

Introduction to the MySQL CHECK constraint

Prior to MySQL 8.0.16, the `CREATE TABLE` allows you to include a table `CHECK` constraint. However, the `CHECK` constraint is just parsed and ignored:

```
CHECK(expression)
```

As of MySQL 8.0.16, the `CREATE TABLE` supported essential features of table and column `CHECK` constraints for all storage engines.

Here is the syntax:

```
[CONSTRAINT [constraint_name]] CHECK (expression) [[NOT] ENFORCED]
```

In this syntax:

First, specify the name for the check constraint that you want to create. If you omit the constraint name, MySQL automatically generates a name with the following convention:

```
table_name_chk_n
```

where `n` is an ordinal number 1,2,3... For example, the names of `CHECK` constraints for the `parts` table will be `parts_chk_1` , `parts_chk_2` , ...

Second, specify a Boolean `expression` which must evaluate to `TRUE` or `UNKNOWN` for each row of the table. If the expression evaluates to `FALSE` , the values violate the constraint or a

constraint violation occurs.

Third, optionally specify enforcement clause to indicate whether the check constraint is enforced:

- Use `ENFORCED` or just omit the `ENFORCED` clause to create and enforce the constraint.
- Use `NOT ENFORCED` to create the constraint but do not enforce it.

As mentioned earlier, you can specify a `CHECK` constraint as a table constraint or column constraint.

A table `CHECK` constraint can reference multiple columns while the column `CHECK` constraint can refer to the only column where it is defined.

MySQL CHECK constraint examples

Let's take some examples of using the `CHECK` constraints.

1) MySQL CHECK constraint – column constraint example

This statement creates a new `parts` table:

```
CREATE TABLE parts (
    part_no VARCHAR(18) PRIMARY KEY,
    description VARCHAR(40),
    cost DECIMAL(10,2 ) NOT NULL CHECK (cost >= 0),
    price DECIMAL(10,2) NOT NULL CHECK (price >= 0)
);
```

In this statement, we have two column `CHECK` constraints: one for the cost column and the other for the price column.

Because we did not explicitly specify the names for the `CHECK` constraints, MySQL automatically generated names for them.

To view the table definition with the `CHECK` constraint name, you use the `SHOW CREATE TABLE` statement:

```
SHOW CREATE TABLE parts;
```

Here is the output:

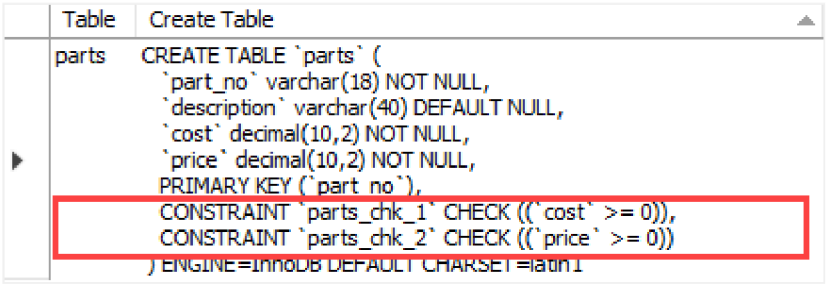

Table	Create Table
parts	CREATE TABLE `parts` (`part_no` varchar(18) NOT NULL, `description` varchar(40) DEFAULT NULL, `cost` decimal(10,2) NOT NULL, `price` decimal(10,2) NOT NULL, PRIMARY KEY (`part_no`), CONSTRAINT `parts_chk_1` CHECK ((`cost` >= 0)), CONSTRAINT `parts_chk_2` CHECK ((`price` >= 0))) ENGINE=InnoDB DEFAULT CHARSET=latin1

As you can see clearly from the output, MySQL generated the check constraint `parts_chk_1` and `parts_chk_2` .

Once the `CHECK` constraints are in place, whenever you insert or update a value that causes the Boolean expression evaluates to false, MySQL rejects the change and issues an error.

This statement inserts a new row into the parts table:

```
INSERT INTO parts(part_no, description,cost,price)
VALUES('A-001','Cooler',0,-100);
```

MySQL issued an error:

```
Error Code: 3819. Check constraint 'parts_chk_2' is violated.
```

Because the value of the `price` column is negative which causes the expression `price > 0` evaluates to `FALSE` that results in a constraint violation.

2) MySQL CHECK constraint – table constraint example

First, drop the `parts` table:

```
DROP TABLE IF EXISTS parts;
```

Then, create a new `parts` table with one more table `CHECK` constraint:

```
CREATE TABLE parts (
    part_no VARCHAR(18) PRIMARY KEY,
    description VARCHAR(40),
    cost DECIMAL(10,2 ) NOT NULL CHECK (cost >= 0),
```

```
    price DECIMAL(10,2) NOT NULL CHECK (price >= 0),
    CONSTRAINT parts_chk_price_gt_cost
        CHECK(price >= cost)
);
```

The following new clause defines a table `CHECK` constraint that ensures the price is always greater than or equal to cost:

```
CONSTRAINT parts_chk_price_gt_cost CHECK(price >= cost)
```

Because we explicitly specify the name for the `CHECK` constraint, MySQL just creates the new constraint with the specified name.

Here is the definition of the `parts` table:

```
SHOW CREATE TABLE parts;
```

Table	Create Table
parts	CREATE TABLE `parts` (`part_no` varchar(18) NOT NULL, `description` varchar(40) DEFAULT NULL, `cost` decimal(10,2) NOT NULL, `price` decimal(10,2) NOT NULL, PRIMARY KEY (`part_no`), CONSTRAINT `parts_chk_1` CHECK ((`cost` >= 0)), CONSTRAINT `parts_chk_2` CHECK ((`price` >= 0)), CONSTRAINT `parts_chk_price_gt_cost` CHECK ((`price` >= `cost`))) ENGINE=InnoDB DEFAULT CHARSET=latin1

The table `CHECK` constraint appears at the end of the table definition after the column list.

This statement attempts to insert a new part whose price is less than cost:

```
INSERT INTO parts(part_no, description,cost,price)
VALUES('A-001','Cooler',200,100);
```

Here is the error due to the constraint violation:

```
Error Code: 3819. Check constraint 'parts_chk_price_gt_cost' is violated.
```

In this tutorial, you have learned about the MySQL `CHECK` constraints to ensure values stored in a column satisfy a Boolean condition.