BITS Pilani, K.K. Birla Goa Campus
DBS Lab 05 Manual
Topics: CASE, IF, VIEW

CASE statement/expression in mysql

CASE statement is a conditional expression. It is used to execute specific statements when some search condition evaluates to TRUE. It can easily check multiple conditions in a single statement.

Use of CASE Expressions in SQL query is sometimes extremely useful. For example, using CASE in SELECT statement provides the developer the power to manipulate the data at presentaion layer without changing data at backend. So there are various use of CASE Expressions and it can be used in, including in

statements like [SELECT, UPDATE, DELETE, SET]
and clauses like [WHERE, ORDER BY, HAVING, GROUP BY]

Here is some more information on CASE expressions

Type of CASE Expression:

- Simple Case Expression
- Searched Case Expression

Basic syntax of CASE expression:

Type-I (Simple Case Expression)

Syntax:

```
CASE InputValue
      WHEN WhenValue THEN ReturnValue
      WHEN WhenValue THEN ReturnValue
      ELSE DefaultReturnValue
END
```

Functionality:

1. In Simple **CASE** Expression, one value is checked against multiple values [Each value at **WHEN** clause]
2. Simple **CASE** Expression only allows equality check.
3. Returns ReturnValue of the first match i.s when InputValue = WhenValue.
4. If no matche is found it returns NULL if ELSE clause is not specified otherwise DefaultReturnValue will be returned.

Note:

1. The DataType of InputValue and WhenValue must be same.

2. The DataType of ReturnValue and DeafaultReturnValue must be same.

3. It can be nested upto 3 levels.

Type-II (Searched Case Expression)

Syntax:

```
CASE
        WHEN BooleanExpression THEN ReturnValue
        WHEN BooleanExpression THEN ReturnValue
        ELSE DefaultReturnValue
END
```

Functionality:

1. In Searched CASE Expression,BooleanExpression will be evaluted for each WHEN clause specified.
2. Returns ReturnValue of the first BooleanExpression evalutes to True.
3. Simple CASE Expression allows comparision using [AND OR] between boolean expression.
4. If no BooleanExpression is evaluted to True then returns NULL if ELSE clause is not specified otherwise DefaultReturnValue will be returned.

Note :
1.The DataType of ReturnValye and DeafaultReturnValue must be same
2.Searched Case Expression have no limit to the number of nesting levels.

Example: Use of CASE with SELECT statement**.**

Simple Case Expression with SELECT statement. (Assume that there is a User table with Name and TypeID columns. We want to print type of different users based on the stored TypeID):

```
SELECT
     Name, CASE TypeID
      WHEN '0' THEN 'Anonymous'
      WHEN '1' THEN 'Registered'
      WHEN '2' THEN 'Admin'
      ELSE NULL
     END AS UserType
FROM
   User;
```

Example: Searched CASE expression with SELECT statement

```
SELECT
     Name, Marks,
     CASE
          WHEN Marks < 350 THEN 'Fail'
          WHEN Marks >= 350 AND Marks < 450 THEN 'THIRD'
          WHEN Marks >=450 AND Marks < 550 THEN 'SECOND'
          WHEN Marks >=550 AND Marks < 650 THEN 'FIRST'
          ELSE 'Excelent'
     END
FROM
```

```
 Student;
```

Demo:

Lets create a STUDENT table.

```
mysql> CREATE TABLE STUDENT( ID DECIMAL PRIMARY KEY, NAME VARCHAR(20),
MARK DECIMAL );
```

Insert a few data into the STUDENT table.

```
mysql> INSERT INTO STUDENT VALUES (1,'ALEX',350),(2,'BIJOY',450),
(3,'CHETAN',100);

mysql> INSERT INTO STUDENT VALUES (4,'DEEPAK',357),(5,'EMINEM',150),
(6,'FOOLY',250);

mysql> INSERT INTO STUDENT VALUES (7,'ARYAN',550),(8,'BISHNU',10),
(9,'SEKHAR',451);

mysql> DESC STUDENT;
+-------+---------------+------+-----+---------+-------+
| Field | Type          | Null | Key | Default | Extra |
+-------+---------------+------+-----+---------+-------+
| ID    | decimal(10,0) | NO   | PRI | NULL    |       |
| NAME  | varchar(20)   | YES  |     | NULL    |       |
| MARK  | decimal(10,0) | YES  |     | NULL    |       |
+-------+---------------+------+-----+---------+-------+

mysql> SELECT * FROM STUDENT;
+----+--------+------+
| ID | NAME   | MARK |
+----+--------+------+
|  1 | ALEX   |  350 |
|  2 | BIJOY  |  450 |
|  3 | CHETAN |  100 |
|  4 | DEEPAK |  357 |
|  5 | EMINEM |  150 |
|  6 | FOOLY  |  250 |
|  7 | ARYAN  |  550 |
|  8 | BISHNU |   10 |
|  9 | SEKHAR |  451 |
+----+--------+------+
```

Let say I want to print division of each student based on above CASE statement.

```
mysql> SELECT ID, NAME, CASE WHEN (MARK*100/600)>=60 THEN 'FIRST'
    -> WHEN (MARK*100/600)>=50 AND (MARK*100/600)<60 THEN 'SECOND'
    -> WHEN (MARK*100/600)>=40 AND (MARK*100/600)<50 THEN 'THIRD'
    -> ELSE 'FAIL' END AS DIVISION
    -> FROM STUDENT;
```

```
+----+--------+----------+
| ID | NAME   | DIVISION |
+----+--------+----------+
|  1 | ALEX   | SECOND   |
|  2 | BIJOY  | FIRST    |
|  3 | CHETAN | FAIL     |
|  4 | DEEPAK | SECOND   |
|  5 | EMINEM | FAIL     |
|  6 | FOOLY  | THIRD    |
|  7 | ARYAN  | FIRST    |
|  8 | BISHNU | FAIL     |
|  9 | SEKHAR | FIRST    |
+----+--------+----------+
9 rows in set (0.03 sec)
```

IF statement

Syntax:
`if(expr1, expr2, expr3)`

If ***expr1*** is TRUE (***expr1*** <> 0 and ***expr1*** <> NULL) then IF() returns ***expr2***; otherwise it returns ***expr3***. IF() returns a numeric or string value, depending on the context in which it is used.
For example:

```
mysql> SELECT IF(1>2,2,3);
+-------------+
| IF(1>2,2,3) |
+-------------+
|           3 |
+-------------+
mysql> SELECT IF(1<2,'yes','no');
+--------------------+
| IF(1<2,'yes','no') |
+--------------------+
| yes                |
+--------------------+

mysql> SELECT IF(STRCMP('test','test1'),'no','yes') RESULT;
+--------+
| RESULT |
+--------+
| no     |
+--------+
```

The default return type of IF() (which may matter when it is stored into a temporary table) is calculated as follows.

Expression	Return Value
expr2 or expr3 returns a string	string
expr2 or expr3 returns a floating-point value	floating-point
expr2 or expr3 returns an integer	integer

Demo:

```
mysql> SELECT ID, NAME, IF(NAME<'F%','GOOD','BAD') REMARK FROM STUDENT
ORDER BY MARK;

+----+--------+------+
| ID | NAME   | NAME |
+----+--------+------+
|  8 | BISHNU | GOOD |
|  3 | CHETAN | GOOD |
|  5 | EMINEM | GOOD |
|  6 | FOOLY  | BAD  |
|  1 | ALEX   | GOOD |
|  4 | DEEPAK | GOOD |
|  2 | BIJOY  | GOOD |
|  9 | SEKHAR | BAD  |
|  7 | ARYAN  | GOOD |
+----+--------+------+

mysql> SELECT ID, NAME, MARK, IF(MARK<300,'BAD','GOOD') REMARK FROM
STUDENT;

+----+--------+------+--------+
| ID | NAME   | MARK | REMARK |
+----+--------+------+--------+
|  1 | ALEX   |  350 | GOOD   |
|  2 | BIJOY  |  450 | GOOD   |
|  3 | CHETAN |  100 | BAD    |
|  4 | DEEPAK |  357 | GOOD   |
|  5 | EMINEM |  150 | BAD    |
|  6 | FOOLY  |  250 | BAD    |
|  7 | ARYAN  |  550 | GOOD   |
|  8 | BISHNU |   10 | BAD    |
|  9 | SEKHAR |  451 | GOOD   |
+----+--------+------+--------+
```

Creating view in mysql

In SQL, a view is a virtual table based on the result-set of an SQL statement. A view contains rows and columns, just like a real table. The fields in a view are fields from one or more real tables in the database.

You can add SQL functions, WHERE, and JOIN statements to a view and present the data as if the data were coming from one single table.

```
CREATE VIEW view_name AS
     SELECT column_name(s)
     FROM table_name
     WHERE condition;
```

Note: A view always shows **up-to-date data.** The database engine recreates the data, using the view's SQL statement, every time a user queries a view.

Demo:

Consider the STUDENT table as discussed above. We can create a view from NAME and MARK column and store it for future purpose.

```
mysql> CREATE VIEW NAME_MARK
     -> AS
     -> SELECT NAME, MARK FROM STUDENT;
Query OK, 0 rows affected (0.04 sec)

mysql> desc NAME_MARK;
+-------+---------------+------+-----+---------+-------+
| Field | Type          | Null | Key | Default | Extra |
+-------+---------------+------+-----+---------+-------+
| NAME  | varchar(20)   | YES  |     | NULL    |       |
| MARK  | decimal(10,0) | YES  |     | NULL    |       |
+-------+---------------+------+-----+---------+-------+
2 rows in set (0.00 sec)

mysql> SELECT * FROM NAME_MARK;
+--------+------+
| NAME   | MARK |
+--------+------+
| ALEX   |  350 |
| BIJOY  |  450 |
| CHETAN |  100 |
| DEEPAK |  357 |
| EMINEM |  150 |
| FOOLY  |  250 |
| ARYAN  |  550 |
| BISHNU |   10 |
| SEKHAR |  451 |
+--------+------+
9 rows in set (0.03 sec)
```

Say after this we inserted two more records to the STUDENT table as follows:

```
mysql> INSERT INTO STUDENT VALUES (10,'NEWSTUD1',355);
Query OK, 1 row affected (0.16 sec)

mysql> INSERT INTO STUDENT VALUES (11,'NEWSTUD2',405);
Query OK, 1 row affected (0.12 sec)
```

Now, let see what the view NAME_MARK contains:

```
mysql> SELECT * FROM NAME_MARK;
+----------+------+
| NAME     | MARK |
+----------+------+
| ALEX     |  350 |
| BIJOY    |  450 |
| CHETAN   |  100 |
| DEEPAK   |  357 |
| EMINEM   |  150 |
| FOOLY    |  250 |
| ARYAN    |  550 |
| BISHNU   |   10 |
| SEKHAR   |  451 |
| NEWSTUD1 |  355 |
| NEWSTUD2 |  405 |
+----------+------+
11 rows in set (0.00 sec)
```

See the two newly inserted records are also coming in the view. But, remember we didn't do anything in the view. It automatically gets updated with regard to the parent table(s).

Lab 06 Evaluation: Based on the concept of CASE, IF and VIEW

----END----