Updating Table Values

The UPDATE statement updates the values in columns of existing rows in the named table with new values.

The portion of the grammar for the UPDATE statement that we will utilize is shown below. For the complete grammar, please see section 13.2.13 UPDATE Statement in the MySQL documentation.

The grammar for table_reference is provided in section 13.2.10.2 JOIN Clause in the MySQL documentation. For our purposes, we will simply use a table's name as the table_reference in all our UPDATE statements.

```
UPDATE table_reference
    SET assignment_list
    [WHERE where_condition]
    [LIMIT row_count]

value:
    {expr | DEFAULT}

assignment:
    col_name = value

assignment_list:
    assignment [, assignment] ...
```

The SET clause indicates which columns to modify and the values they should be given. Each value can be given as an expression or as the DEFAULT keyword to set a column's value explicitly to its default value.

```
SET usedVacationDays=0

SET userName = 'Beavis', highScore = DEFAULT
```

Safe Update Mode

By default, all databases are placed in Safe Update Mode. This mode requires that either

- a WHERE clause that references the PRIMARY KEY column(s) or
- a LIMIT clause be included in every UPDATE statement.

A WHERE clause specifies which rows to alter and a LIMIT clause specifies a maximum number of rows that can be altered.

Without a WHERE or LIMIT clause, all rows in the table would be affected.

To allow all rows to be affected, we need to issue the following statement to turn off the default safe update mode.

```
SET SQL_SAFE_UPDATES=0;
```

This allows us to issue UPDATE commands without a WHERE clause or a LIMIT clause, such as the one below.

```
UPDATE Employee_T
    SET usedVacationDays = 0;
```

With SQL_SAFE_UPDATES turned off, we can also run UPDATE statements that have WHERE clauses, but don't reference the PRIMARY KEY in the WHERE clause, like in the example below.

```
UPDATE Employee_T
    SET maxVacationDays = maxVacationDays + 3
    WHERE yearsOfService = 10;
```

WHERE Clauses

The WHERE clause specifies the conditions that identify which rows to update.

A WHERE clause requires a where_condition. There is no proper grammar for where_condition in all of the MySQL documentation, but a where condition is described as any expression that evaluates to either true or false. See section 9.5 Expressions for the grammar of valid boolean expressions.

Below are the Boolean and relational operators that are used in MySQL.

OR, ||, XOR, AND, &&, NOT, !
=, >= , > , <= , < !=

Note that = is the comparison operator in MySQL, not == like in Jave and c.

Below are some examples of WHERE clauses that contain Boolean and relational operators:

```
WHERE employeeID = 12345

WHERE usedVacationDays < 10 AND yearsOfService > 5

WHERE firstName = 'Barry'
```

WHERE clauses can also contain predicate expressions using the IN, BETWEEN, and LIKE keywords.

For example, we can specify a set of strings and update the rows that have values in a column that are among the set.

```
UPDATE Employee_T
    SET firstName='Lucky'
    WHERE lastName IN ('Buck', 'Yearling');
```

We can use the BETWEEN keyword update rows that have values in a column that are between a lower bound and an upper bound. Note that BETWEEN is inclusive.

```
UPDATE Employee_T
    SET usedVacationDays=usedVacationDays+3
    WHERE yearsOfService BETWEEN 5 AND 10;
```

We can also update rows where the strings in column match a pattern. For example, we can update the rows that have string values in the firstName column that start with B by using the LIKE keyword and the % wildcard character.

```
UPDATE Employee_T
    SET usedVacationDays=1
    WHERE emailAddress LIKE '%.edu';
```