Data Manipulation Language (DML)

DSC 301: Lecture 14

March 26, 2021

Until now we only had *read* privileges e.g., chmod 444). Now we add write privileges (e.g., chmod 666) which enables us to insert, update, and delete data from any (or all) tables. The SQL keywords we examine in this lecture are: `INSERT`, `UPDATE`, and `DELETE`.

Lecture Objectives

This lecture cover the data manipulation language (DML) items:

- `INSERT`
- `UPDATE`
- `DELETE`

`INSERT`

It is obviously necessary to populate the database initially, else we would have no data to query. This data needed to be "inserted into" the database. To **insert** one or more new records **into** a table (e.g., new customer, new product, new categories, etc.), use the following syntax:

```
INSERT [INTO] <TableX>[(column_1,
    column_2,
    column_3,
    .
    .
    .
    column_N)]
VALUES('value_1',
    'value_2',
    .
    .
    .
    'value_N');
```

Insert Rules

1. `INTO` may not be necessary depending on the specific implementation of SQL.
    a. It is good practice to `INTO` to ensure the SQL code is portable to other DBMS.
2. If no column list is specified, the `VALUES` list must contain all columns in the order given by the table.
3. If column allows null values, enter the `NULL` keyword.
4. If the column is defined with a default value (or auto-incremented), then use the `DEFAULT` keyword to assign the default values.
5. If a column list is included (preferred), then columns with default values and null values can be omitted and automatically generated.
6. The column list can be ordered in any way, but the values must then follow this same ordering.
7. Data values must be compatible with the data type specified by the column.
    a. Strings, dates, must contain single quotes, which are optional on numeric literals.
    b. the left quote, ‘, is often used

Example 1. *Insert a new customer into the Customers table.*

```
INSERT INTO Customers(id,
    fname,
    lname,
    phone,
    email,
    address1,
    address2,
    zip)
VALUES(
    default,
    'John',
    'Smith',
    '941-261-9620',
    'smitty@yopmail',
    '810 Hollywood Blvd.',
    '',
    '90210');
```

Example 2. *Insert multiple new customer into the Customers table.*

```
INSERT INTO Customers(id,
    fname,
    lname,
    phone,
    email,
    address1,
    address2,
    zip)
VALUES
    (default, 'John', 'Jones', '207-743-8632', 'jonesy@gmail.com',
        '85 E. 5th St.','Apt. #3','04106'),
    (default, 'Jane', 'Smith', '216-285-9512', 'maryjane@gmail.com'
        , '420 Franklin Ave.','','26034'),
    (default, 'Rick', 'James', '513-392-1128', 'rj@yahoo.com', '
        4927 Norwich Ave.','','04005');
```

UPDATE

The basic syntax for `UPDATE` queries follows (remember statements in square brackets are optional).

```
UPDATE <TableX>
SET <col1>  = expression1 [, <col2> = expression2, ...]
[WHERE <someColumn> = valueX];
```

In particular, the syntax to update a single value is:

```
UPDATE <TableX>
SET <colName>  = value1
WHERE <someColumn> = value2;
```

Notes for `UPDATE`

- The value for a column can be a literal or an expression.
- `WHERE` is used to specify the conditions that must be met for a records to be updated.
- By default, MySQL Workbench runs in *safe mode* that prevents from updateing rows if the `WHERE` clause is omitted or does not refer to a primary key or foreign key column. To circumvent this restriction[1], go to `Edit` → `Preferences` menus and select `SQL Editor` and uncheck "Safe Updates".

[1] **Warning!** Proceed with care, you could update ALL rows in the table without `WHERE`

Another method is to execute `SET SQL_SAFE_UPDATES=0` statement prior to the `UPDATE` query.

- You can `SET`: One column for one row

```
UPDATE TblA SET colX=value1 WHERE colY=value2;
```

- You can `SET`: One column for multiple rows (if `colY=value2` is in multiple records)

```
UPDATE TblA SET colX=value1 WHERE colY=value2;
```

- You can `SET`: Multiple columns for one row (e.g., two column values)

```
UPDATE TblA SET
    colX=value1,
    colY=value2
WHERE colZ=value3;
```

- A subquery can be used in the `WHERE` clause to identify rows to be updated.

```
UPDATE TblA SET
    colX=value1,
    colY=value2
WHERE colZ IN
    (SELECT colA FROM TblB WHERE colB = value3);
```

Example (from Store)

```
UPDATE Products SET
    category_id = 5000
WHERE category_id IN
    (SELECT category_id FROM Categories
    WHERE category like 'Unknown%');
```

DELETE

It is necessary to **delete** records from the database on occasion. NOTE: The is a large potential do damage the database by losing records by performing the delete operation. Therefore, **ALWAYS** use a `WHERE` clause. `DELETE` removes rows, not columns (therefore, you do not list column names in a `DELETE` query.

```
DELETE FROM  <TableX>
WHERE <someColumn> = value;
```

Delete Rules

1. `DELETE` statement is used to delete one or more records from a specified table.
2. Use a `WHERE` clause to specify conditions that must be satisfied for records to be removed[2].
3. SAFE-MODE is automatically ON in MySQL Workbench. Reason: see previous item's footnote.
4. Subqueries can be used within the `WHERE` condition.

```
DELETE FROM Categories WHERE category_id IN
    (
    SELECT category_id
    FROM Categories
    WHERE category LIKE 'Unknow%'
    );
```

5. A foreign key constraint may prevent deletion. Must delete all "child rows" first to prevent orphan records.

Example 3. *Remove the customer 1232.*

```
DELETE FROM  Customers
WHERE id = '1232';
```

---

[2] **Warning!** Always use a `WHERE` clause. Without a `WHERE` condition, ALL records will be deleted.