Chapter 5

How to insert, update, and delete data

Objectives

Applied

1. Create a copy of a table that can be used for testing INSERT, UPDATE, and DELETE statements.
2. Given the specifications for an action that modifies data, code the INSERT, UPDATE, or DELETE statement for doing the action.

Knowledge

1. Describe MySQL's default behavior when you execute an INSERT, UPDATE, or DELETE statement.
2. Explain how to handle null values and default values when coding INSERT and UPDATE statements.

The syntax of the CREATE TABLE AS statement

```
CREATE TABLE table_name AS select_statement
```

Create a complete copy of the Invoices table

```
CREATE TABLE invoices_copy AS
SELECT *
FROM invoices
```

Create a partial copy of the Invoices table

```
CREATE TABLE old_invoices AS
SELECT *
FROM invoices
WHERE invoice_total - payment_total - credit_total = 0
```

Create a table with summary rows from the Invoices table

```
CREATE TABLE vendor_balances AS
SELECT vendor_id, SUM(invoice_total) AS sum_of_invoices
FROM invoices
WHERE (invoice_total - payment_total - credit_total) <> 0
GROUP BY vendor_id
```

Delete a table

```
DROP TABLE old_invoices
```

The syntax of the INSERT statement

```
INSERT [INTO] table_name [(column_list)]
VALUES (expression_1[, expression_2]...)[,
       (expression_1[, expression_2]...)]...
```

The column definitions for the Invoices table

```
invoice_id              INT             NOT NULL
                        AUTO_INCREMENT,
vendor_id               INT             NOT NULL,
invoice_number          VARCHAR(50)     NOT NULL,
invoice_date            DATE            NOT NULL,
invoice_total           DECIMAL(9,2)    NOT NULL,
payment_total           DECIMAL(9,2)    NOT NULL
                        DEFAULT 0,
credit_total            DECIMAL(9,2)    NOT NULL
                        DEFAULT 0,
terms_id                INT             NOT NULL,
invoice_due_date        DATE            NOT NULL,
payment_date            DATE
```

Insert a single row without using a column list

```
INSERT INTO invoices VALUES
(115, 97, '456789', '2018-08-01', 8344.50, 0, 0, 1,
'2018-08-31', NULL)

(1 row affected)
```

Insert a single row using a column list

```
INSERT INTO invoices
    (vendor_id, invoice_number, invoice_total, terms_id,
    invoice_date, invoice_due_date)
VALUES
    (97, '456789', 8344.50, 1, '2018-08-01',
    '2018-08-31')

(1 row affected)
```

Insert multiple rows

```
INSERT INTO invoices VALUES
    (116, 97, '456701', '2018-08-02', 270.50, 0, 0, 1,
    '2018-09-01', NULL),
    (117, 97, '456791', '2018-08-03', 4390.00, 0, 0, 1,
    '2018-09-02', NULL),
    (118, 97, '456792', '2018-08-03', 565.60, 0, 0, 1,
    '2018-09-02', NULL)

(3 rows affected)
```

The column definitions for the Color_Sample table

```
color_id        INT             NOT NULL      AUTO_INCREMENT,
color_number    INT             NOT NULL      DEFAULT 0,
color_name      VARCHAR(50)
```

INSERT statements for the Color_Sample table

```
INSERT INTO color_sample (color_number)
VALUES (606)

INSERT INTO color_sample (color_name)
VALUES ('Yellow')

INSERT INTO color_sample
VALUES (DEFAULT, DEFAULT, 'Orange')

INSERT INTO color_sample
VALUES (DEFAULT, 808, NULL)

INSERT INTO color_sample
VALUES (DEFAULT, DEFAULT, NULL)
```

The Color_Sample table with the inserted rows

color_id	color_number	color_name
1	606	NULL
2	0	Yellow
3	0	Orange
4	808	NULL
5	0	NULL

The syntax for using a subquery to insert one or more rows

```
INSERT [INTO] table_name [(column_list)]
select_statement
```

Insert paid invoices into the Invoice_Archive table

```
INSERT INTO invoice_archive
SELECT *
FROM invoices
WHERE invoice_total - payment_total - credit_total = 0

(103 rows affected)
```

The same statement with a column list

```
INSERT INTO invoice_archive
    (invoice_id, vendor_id, invoice_number,
    invoice_total, credit_total, payment_total,
    terms_id, invoice_date, invoice_due_date)
SELECT
    invoice_id, vendor_id, invoice_number,
    invoice_total, credit_total, payment_total,
    terms_id, invoice_date, invoice_due_date
FROM invoices
WHERE invoice_total - payment_total - credit_total = 0

(103 rows affected)
```

The syntax of the UPDATE statement

```
UPDATE table_name
SET column_name_1 = expression_1
 [, column_name_2 = expression_2]...
[WHERE search_condition]
```

Update two columns for a single row

```
UPDATE invoices
SET payment_date = '2018-09-21',
    payment_total = 19351.18
WHERE invoice_number = '97/522'

(1 row affected)
```

Update one column for multiple rows

```
UPDATE invoices
SET terms_id = 1
WHERE vendor_id = 95

(6 rows affected)
```

Update one column for one row

```
UPDATE invoices
SET credit_total = credit_total + 100
WHERE invoice_number = '97/522'

(1 row affected)
```

Safe update mode in MySQL Workbench

- By default, MySQL Workbench runs in safe update mode.
- Safe update mode prevents you from updating rows if the WHERE clause is omitted or doesn't refer to a primary key or foreign key column.
- You can turn safe update mode off by selecting the Edit→Preferences command, selecting the SQL Editor node, changing the "Safe Updates" option, and restarting MySQL Workbench.

Warning

- If you turn off safe update mode and omit the WHERE clause, all rows in the table will be updated.

Update all invoices for a vendor

```
UPDATE invoices
SET terms_id = 1
WHERE vendor_id =
   (SELECT vendor_id
    FROM vendors
    WHERE vendor_name = 'Pacific Bell')
(6 rows affected)
```

Update the terms for all invoices for vendors in three states

```
UPDATE invoices
SET terms_id = 1
WHERE vendor_id IN
   (SELECT vendor_id
    FROM vendors
    WHERE vendor_state IN ('CA', 'AZ', 'NV'))
(40 rows affected)
```

The syntax of the DELETE statement

```
DELETE FROM table_name
[WHERE search_condition]
```

Delete one row

```
DELETE FROM general_ledger_accounts
WHERE account_number = 306

(1 row affected)
```

Delete one row using a compound condition

```
DELETE FROM invoice_line_items
WHERE invoice_id = 78 AND invoice_sequence = 2

(1 row affected)
```

Delete multiple rows

```
DELETE FROM invoice_line_items
WHERE invoice_id = 12

(4 rows affected)
```

Use a subquery in a DELETE statement

```
DELETE FROM invoice_line_items
WHERE invoice_id IN
    (SELECT invoice_id
     FROM invoices
     WHERE vendor_id = 115)

(4 rows affected)
```

Warning

- If you turn safe update mode off and omit the WHERE clause from a DELETE statement, all the rows in the table will be deleted.