4.10 Inserting, updating, and deleting rows

INSERT statement

The ***INSERT*** statement adds rows to a table. The INSERT statement includes the INTO and VALUES clauses:

- The ***INTO*** clause names the table and columns where data is to be added.
- The ***VALUES*** clause specifies the column values to be added.

The VALUES clause may list any number of rows in parentheses to insert multiple rows.

Figure 4.10.1: INSERT syntax.

```
INSERT INTO TableName (Column1, Column2, ..., ColumnN)
VALUES (Value1, Value2, ..., ValueN);
```

Feedback?

PARTICIPATION ACTIVITY 4.10.1: Inserting rows into the Employee table.

1 2 3 4 2x speed

table name column names

```
INSERT INTO Employee (ID, Name, Salary)
VALUES (2538, 'Lisa Ellison', 45000);
```

Employee

ID	Name	Salary
2538	Lisa Ellison	45000

column values

```
INSERT INTO Employee
VALUES (5384, 'Sam Snead', 30500);


INSERT INTO Employee
VALUES (6381, 'Maria Rodriguez', 92300),
       (7920, 'Jiho Chen', 56000),
       (9385, 'Alexis Parsons', 32000);
```

5384	Sam Snead	30500
6381	Maria Rodriguez	92300
7920	Jiho Chen	56000
9385	Alexis Parsons	32000

Any number of rows may be added with a single INSERT statement.

Captions

1. The INSERT statement's INTO clause names the Employee table and Employee's columns in parentheses.
2. The VALUES clause names the column values in parenthesis. The value order must match the column order in the INTO clause.
3. The column names may be omitted as long as the VALUES clause lists all column values in the same order as the table's columns.
4. Any number of rows may be added with a single INSERT statement.

Feedback?

PARTICIPATION ACTIVITY 4.10.2: INSERT statement.

Refer to the table definition below.

```
CREATE TABLE Department (
   Code TINYINT UNSIGNED,
   Name VARCHAR(20),
   ManagerID SMALLINT UNSIGNED
);
```

1) Which statement correctly inserts Engineering?

- `INSERT INTO Department Code, Name, ManagerID VALUES (44, 'Engineering', 2538);`
- `INSERT INTO (Code, Name, ManagerID) VALUES (44, 'Engineering', 2538);`
- `INSERT INTO Department (Code, Name, ManagerID) VALUES (44, 'Engineering', 2538);`

Correct

The INSERT INTO clause names the table and columns, and the VALUES clause names the column values.

2) Which statement correctly inserts Sales?

- `INSERT Department VALUES (82, 'Sales', 6381);`
- `INSERT INTO Department VALUES ('Sales', 82, 6381);`
- `INSERT INTO Department (82, 'Sales', 6381);`

Correct

The keyword INTO and column names are optional and may be omitted. When column names are omitted, values must be in the order of columns in the CREATE TABLE statement.

3) Which statement correctly inserts Marketing?

- `INSERT INTO Department (Name, ManagerID, Code) VALUES (12, 'Marketing', 6381);`

Correct

The columns values must match the order of the column names in the INSERT INTO clause.

```
INSERT INTO Department
(Name, ManagerID, Code)
VALUES ('Marketing',
6381, 12);
```

```
INSERT INTO Department
(Name, ManagerID, Code)
VALUES (6381, 12,
'Marketing');
```

4) Which statement correctly inserts Technical Support?

```
INSERT INTO Department
(Code, Name)
VALUES (99);
```

```
INSERT INTO Department
(Code, Name)
VALUES (99, 'Technical
Support');
```

```
INSERT INTO Department
(Code, Name)
VALUES (99, 'Technical
Support', NULL);
```

Correct

The INSERT INTO clause lists Code and Name, and the VALUES clause lists the two matching values. The unspecified ManagerID value is NULL.

Feedback?

Common INSERT errors

Database users make some common errors when creating INSERT statements:

- Inserting duplicate primary key values or foreign key values that do not match an existing primary key.
- Inserting primary key values for auto-increment columns.
- Inserting NULL values for columns that are NOT NULL.

PARTICIPATION ACTIVITY 4.10.3: Common INSERT errors.

1 2 3 2x speed

✗
```
INSERT INTO Employee
VALUES (2538, 'Maria Rodriguez', 92300);
```

✓
```
INSERT INTO Employee
VALUES (6381, 'Maria Rodriguez', 92300);
```

Employee

ID	Name	Salary
2538	Lisa Ellison	45000
5384	Sam Snead	30500
6381	Maria Rodriguez	92300

✗
```
INSERT INTO Employee
VALUES (3, 'Maria Rodriguez', 92300);
```

✓
```
INSERT INTO Employee (Name, Salary)
VALUES ('Maria Rodriguez', 92300);
```

auto-increment

Employee

ID	Name	Salary
1	Lisa Ellison	45000
2	Sam Snead	30500
3	Maria Rodriguez	92300

✗
```
INSERT INTO Employee (ID, Name)
VALUES (6381, 'Maria Rodriguez');
```

✓
```
INSERT INTO Employee (ID, Name, Salary)
VALUES (6381, 'Maria Rodriguez', 0);
```

Employee

NOT NULL

ID	Name	Salary
2538	Lisa Ellison	45000
5384	Sam Snead	30500
6381	Maria Rodriguez	0

If Salary is a NOT NULL column, then the Salary value must be specified.

Captions

1. The INSERT statement uses an ID that already exists in Employee. Duplicate primary key values cannot be added, so a unique ID must be chosen.

2. If ID is an auto-increment column, the ID should not be listed in the INSERT statement. The database assigns the ID automatically.
3. If Salary is a NOT NULL column, then the Salary value must be specified.

Feedback?

PARTICIPATION ACTIVITY 4.10.4: Insert rows into Movie table.

The given SQL creates a Movie table with an auto-incrementing ID column.

Write a single INSERT statement immediately after the CREATE TABLE statement that inserts the following movies:

Title	Rating	Release Date
Raiders of the Lost Ark	PG	June 15, 1981
The Godfather	R	March 24, 1972
The Pursuit of Happyness	PG-13	December 15, 2006

Note that dates above need to be converted into 'YYYY-MM-DD' format in the INSERT statement.

Run your solution and verify the movies in the result table have the auto-assigned IDs 1, 2, and 3.

```
1
2  CREATE TABLE Movie (
3    ID INT AUTO_INCREMENT,
4    Title VARCHAR(100),
5    Rating CHAR(5) CHECK (Rating IN ('G', 'PG', 'PG-13', 'R')),
6    ReleaseDate DATE,
7    PRIMARY KEY (ID)
8  );
9
10 -- Write your INSERT statement here:
```

```
11
12
13
14 SELECT *
15 FROM Movie;
16
```

Run Reset code

Feedback?

PARTICIPATION ACTIVITY 4.10.5: Common INSERT errors.

Refer to the column information produced by the statement `SHOW COLUMNS FROM Department`. ManagerID is a foreign key that references the Employee ID column. The ManagerID's Key value "MUL" means the ManagerID may have multiple repeating values. Ex: Two different departments may have the same manager.

Result

Field	Type	Null	Key	Default	Extra
Code	tinyint(3) unsigned	NO	PRI	NULL	auto_increment
Name	varchar(20)	NO		NULL	
ManagerID	smallint(5) unsigned	YES	MUL	NULL	

1) Which statement correctly inserts Engineering?

- ```
  INSERT INTO Department
  (Code, Name, ManagerID)
  VALUES (44,
  'Engineering', 2538);
  ```
- ```
  INSERT INTO Department
  VALUES ('Engineering',
  2538);
  ```

Correct

The Code column is auto-increment, so the database chooses a unique integer for Code.

- ◉ `INSERT INTO Department (Name, ManagerID) VALUES ('Engineering', 2538);`

2) Which statement correctly inserts Sales if the only Employee IDs are 2538 and 6381?

- ○ `INSERT INTO Department (Name, ManagerID) VALUES (2538, 'Sales');`
- ○ `INSERT INTO Department (Name, ManagerID) VALUES ('Sales', 0202);`
- ◉ `INSERT INTO Department (Name, ManagerID) VALUES ('Sales', 6381);`

Correct

The foreign key value 6381 matches the existing primary key value 6381.

3) Which statement correctly inserts an unnamed department with no manager?

- ◉ `INSERT INTO Department (Name, ManagerID) VALUES ('', NULL);`
- ○ `INSERT INTO Department (Name, ManagerID) VALUES (NULL, NULL);`
- ○ `INSERT INTO Department (Name, ManagerID) VALUES ('');`

Correct

The SHOW COLUMNS result table shows the Name column does not allow NULL values. The Name value is specified as an empty string, which is allowed. The ManagerID can be NULL, which indicates no manager is assigned to the department.

Feedback?

UPDATE statement

The ***UPDATE*** statement modifies existing rows in a table. The UPDATE statement uses the ***SET*** clause to specify the new column values.

The UPDATE statement uses a WHERE clause to determine which rows are updated. The ***WHERE*** clause is used with UPDATE, DELETE, and SELECT statements to specify a condition that must be true for a row to be chosen. Omitting the WHERE clause results in all rows being updated.

Figure 4.10.2: UPDATE syntax.

```
UPDATE TableName
SET Column1 = Value1, Column2 = Value2, ..., ColumnN = ValueN
WHERE condition;
```

Feedback?

PARTICIPATION ACTIVITY 4.10.6: Updating rows in the Employee table.

1 2 2x speed

```
UPDATE Employee
SET Name = 'Tom Snead',
    BirthDate = '2000-03-15'
WHERE ID = 5384;

UPDATE Employee
SET Salary = 42000;
```

Employee

ID	Name	BirthDate	Salary
2538	Lisa Ellison	1993-10-02	42000
5384	Tom Snead	2000-03-15	42000
6381	Maria Rodriguez	2001-12-21	42000

An UPDATE statement with no WHERE clause changes all the rows. Everyone is given a 42,000 salary.

Captions

1. The UPDATE clause indicates the Employee table will be changed. The SET clause names the new Name and BirthDate values. The WHERE clause indicates that only the row with ID 5384 will be changed.
2. An UPDATE statement with no WHERE clause changes all the rows. Everyone is given a 42,000 salary.

Feedback?

PARTICIPATION ACTIVITY 4.10.7: Update the Song table.

The given SQL creates a Song table and inserts three songs.

Write three UPDATE statements to make the following changes:

- Change the title from 'One' to 'With Or Without You'.
- Change the artist from 'The Righteous Brothers' to 'Aretha Franklin'.
- Change the release years of all songs after 1990 to 2021.

Run your solution and verify the songs in the result table reflect the changes above.

```
CREATE TABLE Song (
  ID INT,
  Title VARCHAR(60),
  Artist VARCHAR(60),
  ReleaseYear INT,
  PRIMARY KEY (ID)
);

INSERT INTO Song VALUES
  (100, 'Blinding Lights', 'The Weeknd', 2019),
  (200, 'One', 'U2', 1991),
  (300, 'You\'ve Lost That Lovin\' Feeling', 'The Righteous Brothers', 1964),
  (400, 'Johnny B. Goode', 'Chuck Berry', 1958);
```

```
14
15 -- Write your UPDATE statements here:
16
17
18
19 SELECT *
20 FROM Song;
```

Run Reset code

Feedback?

PARTICIPATION ACTIVITY 4.10.8: UPDATE statement.

Refer to the Department table.

Department

•Code	Name	ManagerID
44	Engineering	2538
82	Sales	6381
12	Marketing	6381
99	Technical support	NULL

1) What is missing to change "Sales" to "Custodial"?

```
UPDATE Department
SET ___
WHERE Code = 82;
```

- ○ 'Sales' = 'Custodial'
- ○ Department = 'Custodial'
- ◉ Name = 'Custodial'

Correct

The SET clause names the column to change and the new value.

2) What departments are changed?

```
UPDATE Department
SET Name = 'Administration'
WHERE ManagerID IS NOT NULL;
```

- ○ Administration
- ◉ All departments except technical support
- ○ All departments

Correct

The WHERE clause changes only rows that do not have a NULL manager. Only technical support has a NULL manager, so all other departments are renamed "Administration".

3) What is missing to change Marketing's Code to 55 and Name to "Administration"?

```
UPDATE Department
SET Code = 55, Name =
'Administration'
WHERE ___;
```

- ○ Code = 55
- ○ ManagerID = 6381
- ◉ Code = 12

Correct

The Marketing department's Code is 12, so only the Marketing department's row is changed.

4) What is Engineering's Code changed to?

```
UPDATE Department
SET Code = 82
WHERE Name = 'Engineering';
```

- ○ 82
- ○ NULL
- ◉ No change

Correct

The Code cannot be changed to 82 because Sales' Code is 82, and all primary key values must be unique. The UPDATE is rejected.

5) What department managers are changed?

Correct

```
UPDATE Department
SET ManagerID = 2538;
```

No WHERE clause exists, so all departments are assigned the same manager.

- (●) All managers
- ( ) No managers
- ( ) Only the Engineering manager

Feedback?

DELETE statement

The ***DELETE*** statement deletes existing rows in a table. The ***FROM*** keyword is followed by the table name whose rows are to be deleted. The WHERE clause specifies which rows should be deleted. Omitting the WHERE clause results in all rows in the table being deleted.

Figure 4.10.3: DELETE syntax.

```
DELETE FROM TableName
WHERE condition;
```

Feedback?

PARTICIPATION ACTIVITY 4.10.9: Deleting rows from the Employee table.

Start ☐ 2x speed

Employee

```
DELETE FROM Employee
WHERE ID = 6381;
```

ID	Name	BirthDate	Salary
2538	Lisa Ellison	1993-10-02	45000
5384	Sam Snead	1995-03-15	30500
~~6381~~	~~Maria Rodriguez~~	~~2001-12-21~~	~~72300~~

```
DELETE FROM Employee
WHERE Salary > 40000 AND
      Salary < 80000;
```

Employee

● ID	Name	BirthDate	Salary
~~2538~~	~~Lisa Ellison~~	~~1993-10-02~~	~~45000~~
5384	Sam Snead	1995-03-15	30500
~~6381~~	~~Maria Rodriguez~~	~~2001-12-21~~	~~72300~~

```
DELETE FROM Employee;
```

Employee

● ID	Name	BirthDate	Salary
~~2538~~	~~Lisa Ellison~~	~~1993-10-02~~	~~45000~~
~~5384~~	~~Sam Snead~~	~~1995-03-15~~	~~30500~~
~~6381~~	~~Maria Rodriguez~~	~~2001-12-21~~	~~72300~~

Captions

1. The DELETE statement deletes the row with ID 6381 from the Employee table.
2. The DELETE statement deletes rows where the Salary is between 40,000 and 80,000. Lisa and Maria are deleted.
3. The DELETE statement has no WHERE clause, so all employees are deleted.

Feedback?

PARTICIPATION ACTIVITY 4.10.10: DELETE statement.

Refer to the Department table.

Department

●Code	Name	ManagerID
44	Engineering	2538
82	Sales	6381

•Code	Name	ManagerID
12	Marketing	6381
99	Technical support	NULL

1) What departments are deleted?

```
DELETE FROM Department
WHERE ManagerID = 6381;
```

- ○ Sales only
- ○ Marketing only
- ◉ Sales and Marketing

Correct

Two rows have ManagerID 6381, so both rows are deleted.

2) What departments are deleted?

```
DELETE FROM Department;
```

- ◉ All departments
- ○ No departments
- ○ Engineering only

Correct

No WHERE clause is specified, so all rows are deleted.

3) What is missing to delete only Sales?

```
DELETE FROM Department
WHERE ___;
```

- ○ ManagerID = 6381
- ◉ Code = 82
- ○ Code = 'Sales'

Correct

Only the Sales row has Code 82, so only the Sales row is deleted.

Feedback?

TRUNCATE

TRUNCATE

*The **TRUNCATE** statement deletes all rows from a table. TRUNCATE is nearly identical to a DELETE statement with no WHERE clause except for some small differences that depend on the database system. Ex: In MySQL, A TRUNCATE statement resets the table's auto-increment values back to 1, but a DELETE statement does not.*

```
TRUNCATE TABLE TableName;
```

Exploring further:

- INSERT Syntax from MySQL.com
- UPDATE Syntax from MySQL.com
- DELETE Syntax from MySQL.com

CHALLENGE ACTIVITY 4.10.1: Inserting, updating, and deleting rows.

379958.2369558.qx3zqy7

Start

1 2 3 4

Refer to the column information produced by the `SHOW COLUMNS FROM Country;` statement.

Result

Field	Type	Null	Key	Default	Extra
ISOCode2	char(2)	NO	PRI	NULL	
Population	integer	YES		0	
Name	varchar(50)	NO		NULL	

Complete the statement to correctly insert a row with ISOCode2 'SD', Name 'Sudan', and Popu
Write the most straightforward code.

```
INSERT INTO Country (ISOCode2, Population, Name)
/* Your code here */ ;
```

1 2 3 4

Check Next

Feedback?

How was this section? Provide feedback