UPDATE to Edit Row

by Sophia

WHAT'S COVERED

In this lesson, you will use the UPDATE statement to update data in a single row, in three parts. Specifically, this lesson will cover:

1. The UPDATE Statement
2. The RETURNING Clause
3. ROLLBACK and SAVEPOINT

1. The UPDATE Statement

The **UPDATE** statement, as the name implies, updates the values in one or more columns of a table in the rows that meet the specified criteria.

The syntax for UPDATE looks like the following:

```
UPDATE <tablename>
SET <column1> = <value1>, <column2> = <value2>, ....
WHERE <condition>;
```

First, we need to specify the table name that we will be updating. Then, in the SET clause, we need to specify the columns and values they will be set to. Any columns in the table that are not specified in the SET clause will just keep the values that they currently have.

The condition in the WHERE clause determines which rows to update. If we do not set a WHERE clause, the UPDATE statement will run on all rows in the table, which might change data that we do not want to be changed. We want to ensure that we are adding the WHERE clause in the UPDATE statement. The structure of the WHERE clause in the UPDATE statement should be very familiar to you, as it is the same structure you have used in the SELECT statement and will use in the DELETE statement.

Let's revisit a scenario that we looked at in a prior lesson. Let's rebuild that referral table and insert the data:

```
CREATE TABLE referral( referral_id SERIAL, first_name VARCHAR(50), last_name VARCHAR(50), email VARCHAR(50) );
INSERT INTO referral (first_name,last_name,email)
VALUES ('Sandra','Boynton','s.boy@email.com'),
('Randall','Faustino','s.boy@email.com'),
('Park','Deanna','p.deanna@email.com'),
('Sunil','Carrie','s.carrie@email.com'),
('Jon','Brianna','j.brianna@email.com'),
('Lana','Jakoba','l.jakoba@email.com'),
('Tiffany','Walker','t.walk@email.com');
```

As a reminder, the issue was that Randall Faustino has the same email as Sandra Boynton, when Randall's email should be r.faustino@email.com. This is where we can do our UPDATE command to fix the email address:

Query Results

Row count: 7

referral_id	first_name	last_name	email
1	Sandra	Boynton	s.boy@email.com
2	Randall	Faustino	s.boy@email.com
3	Park	Deanna	p.deanna@email.com
4	Sunil	Carrie	s.carrie@email.com
5	Jon	Brianna	j.brianna@email.com
6	Lana	Jakoba	l.jakoba@email.com
7	Tiffany	Walker	t.walk@email.com

We could update this by doing the following:

```
UPDATE referral
SET email = 'r.faustino@email.com'
WHERE first_name = 'Randall';
```

However, to ensure that we are updating only Randall Faustino's record, it is ideal to update it using the primary key. There might be more than one Randall in the database, so using the primary key is going to make sure just Randall Faustino is updated correctly. While in our case, we only have a single record with Randall Faustino, in a real-world scenario, we may have multiple Randalls as the first name and potentially multiple Randall Faustinos as well. If we update it using the primary key, we can ensure that we are only updating that single row:

```
UPDATE referral
SET email = 'r.faustino@email.com'
WHERE referral_id = 2;
```

Query Results

Row count: 7

referral_id	first_name	last_name	email
1	Sandra	Boynton	s.boy@email.com
2	Randall	Faustino	r.faustino@email.com
3	Park	Deanna	p.deanna@email.com
4	Sunil	Carrie	s.carrie@email.com
5	Jon	Brianna	j.brianna@email.com
6	Lana	Jakoba	l.jakoba@email.com
7	Tiffany	Walker	t.walk@email.com

We can also update multiple columns at the same time. For example, if we have a customer (Frank Harris) that has moved to Orlando, we may need to update his address, city, state, and postal_code if he kept the rest of his information the same:

```
SELECT *
FROM customer
WHERE customer_id = 16;
```

Query Results

Row count: 1

customer_id	first_name	last_name	company	address	city	state	country	postal_code	phone	fax	email	support_rep_id
16	Frank	Harris	Google Inc.	1600 Amphitheatre Parkway	Mountain View	CA	USA	94043-1351	+1 (650) 253-0000	+1 (650) 253-0000	fharris@google.com	4

```
UPDATE customer
SET address = '555 International Parkway', city = 'Orlando',state = 'FL', postal_code = '33133-1111'
WHERE customer_id = 16;
```

Notice the UPDATE statement's and the SELECT statement's similarities with the WHERE clause. In making the update, we should see Frank Harris's information change:

Query Results

Row count: 1

customer_id	first_name	last_name	company	address	city	state	country	postal_code	phone	fax	email	support_rep_id
16	Frank	Harris	Google Inc.	555 International Parkway	Orlando	FL	USA	33133-1111	+1 (650) 253-0000	+1 (650) 253-0000	fharris@google.com	4

TERM TO KNOW

UPDATE

A statement that updates the content of one or more columns for one or more rows.

2. The RETURNING Clause

Similar to the INSERT statement, the UPDATE statement also has a **RETURNING clause** in PostgreSQL. This returns the updated rows:

```
UPDATE referral
SET email = 'r.faustino@email.com'
WHERE referral_id = 2
RETURNING *;
```

This enables us to validate the affected rows quickly:

Query Results

Row count: 1

referral_id	first_name	last_name	email
2	Randall	Faustino	r.faustino@email.com

```
For example, if we accidentally forgot the WHERE clause:
UPDATE referral
SET email = 'r.faustino@email.com'
RETURNING *;
```

We would be able to see that all of the rows were affected and now had the incorrect email address:

Query Results

Row count: 7

referral_id	first_name	last_name	email
1	Sandra	Boynton	r.faustino@email.com
3	Park	Deanna	r.faustino@email.com
4	Sunil	Carrie	r.faustino@email.com
5	Jon	Brianna	r.faustino@email.com
6	Lana	Jakoba	r.faustino@email.com
7	Tiffany	Walker	r.faustino@email.com
2	Randall	Faustino	r.faustino@email.com

TERM TO KNOW

RETURNING Clause

A clause that can be added to an UPDATE statement to return the updated rows so the results can be inspected.

3. ROLLBACK and SAVEPOINT

Making updates to a database is fraught with potential hazards. To make sure that your updates do not negatively impact the database, you may want to work within a transaction. A **transaction** is a set of steps that are tentatively made but not finalized until you commit the transaction.

To begin a transaction, run the BEGIN statement:

```
BEGIN;
```

Then perform the various operations like inserting, updating, or deleting data.

After performing the operations, you have two options:

- If everything is successful and you want to make the changes permanent, use the COMMIT statement:

```
COMMIT;
```

- If you want to discard the changes made in the transaction, use the **ROLLBACK** statement:

```
ROLLBACK;
```

This will undo all changes made in the current transaction, reverting the database to its state before the BEGIN statement.

A savepoint is a marked point within a transaction to which you can later roll back. Savepoints are useful for creating nested transactions, or for creating a point to which you can roll back without rolling back the entire transaction.

After starting a transaction with BEGIN, use the **SAVEPOINT** statement to create a point:

```
SAVEPOINT my_savepoint;
```

Then perform the operations within the transaction. If you want to roll back to the savepoint, use the ROLLBACK TO statement:

```
ROLLBACK TO my_savepoint;
```

If you no longer need a savepoint and want to commit the changes made since that savepoint, you can use **RELEASE**:

```
RELEASE my_savepoint.
```

Don't forget to COMMIT or ROLLBACK the entire transaction when it has been completed.

Remember that savepoints are only valid within the scope of the current transaction. Once the transaction is committed or rolled back, savepoints are automatically released.

Your turn! Open the SQL tool by clicking on the LAUNCH DATABASE button below. Then, enter in one of the examples above and see how it works. Next, try your own choices for which columns you want the query to provide.

ROLLBACK

A statement that enables you to undo a transaction.

SAVEPOINT

A statement that creates a saved point in a transaction that you can roll back to with the ROLLBACK TO statement.

RELEASE

A statement that enables you to release a savepoint that is no longer needed.

Transaction

A set of steps that add, modify, or delete data in a database.

SUMMARY

In this lesson, you learned how to use **the UPDATE statement** to modify the values in one or more columns and one or more rows in a table. You also learned that **the RETURNING clause** added to an UPDATE statement results in the display of the affected records automatically after the update is made. You learned to create a transaction using BEGIN, and then to set **savepoints** in it to which you can roll back parts of the transaction. You can then use **ROLLBACK** to undo the entire transaction or ROLLBACK TO to undo back to a particular savepoint.

Source: THIS TUTORIAL WAS AUTHORED BY DR. VINCENT TRAN, PHD (2020) AND Faithe Wempen (2024) FOR SOPHIA LEARNING. PLEASE SEE OUR TERMS OF USE.

TERMS TO KNOW

RELEASE

A statement that enables you to release a savepoint that is no longer needed.

RETURNING Clause

A clause that can be added to an UPDATE statement to return the updated rows so the results can be inspected.

ROLLBACK

A statement that enables you to undo a transaction.

SAVEPOINT

A statement that creates a saved point in a transaction that you can roll back to with the ROLLBACK TO statement.

Transaction

A set of steps that add, modify, or delete data in a database.

UPDATE

A statement that updates the content of one or more columns for one or more rows.