UPDATE to Edit Row

by Sophia Tutorial

WHAT'S COVERED

This tutorial explores using the UPDATE statement to update data in a single row in two parts:

1. UPDATE a Single Row
2. RETURNING Clause

1. UPDATE a Single Row

The UPDATE statement is used to modify data that is already in a table. The syntax 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 what the columns are and what 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.

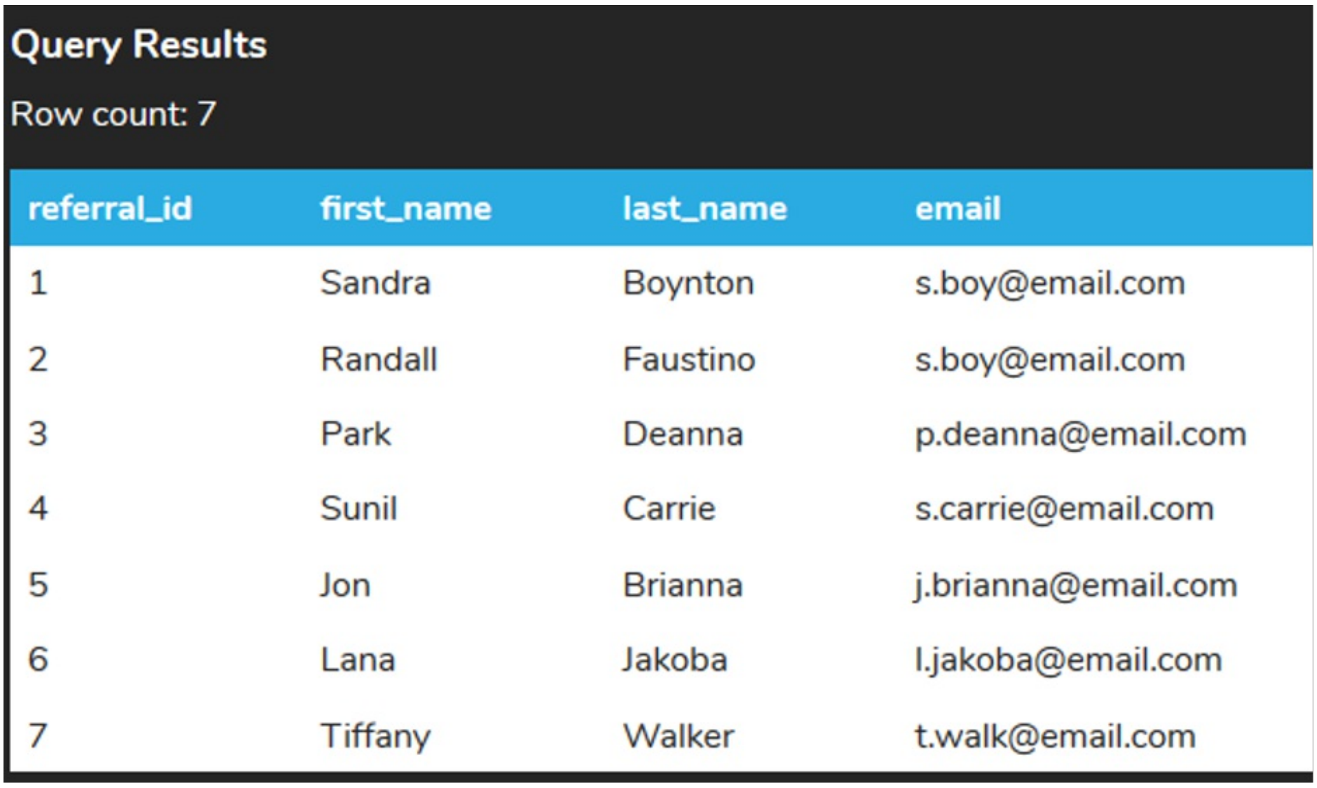
The condition in the WHERE clause determines which rows to update. If we do not set a WHERE clause, the UPDATE statement will run on all rows in the table. In most cases, this is not what we want to do, so we want to ensure that we are adding the WHERE clause in the UPDATE statement. Luckily, the structure of the WHERE clause in the UPDATE statement should be very familiar to you, as it is the same structure as 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 tutorial. 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 instead:


Query Results

Row count: 7

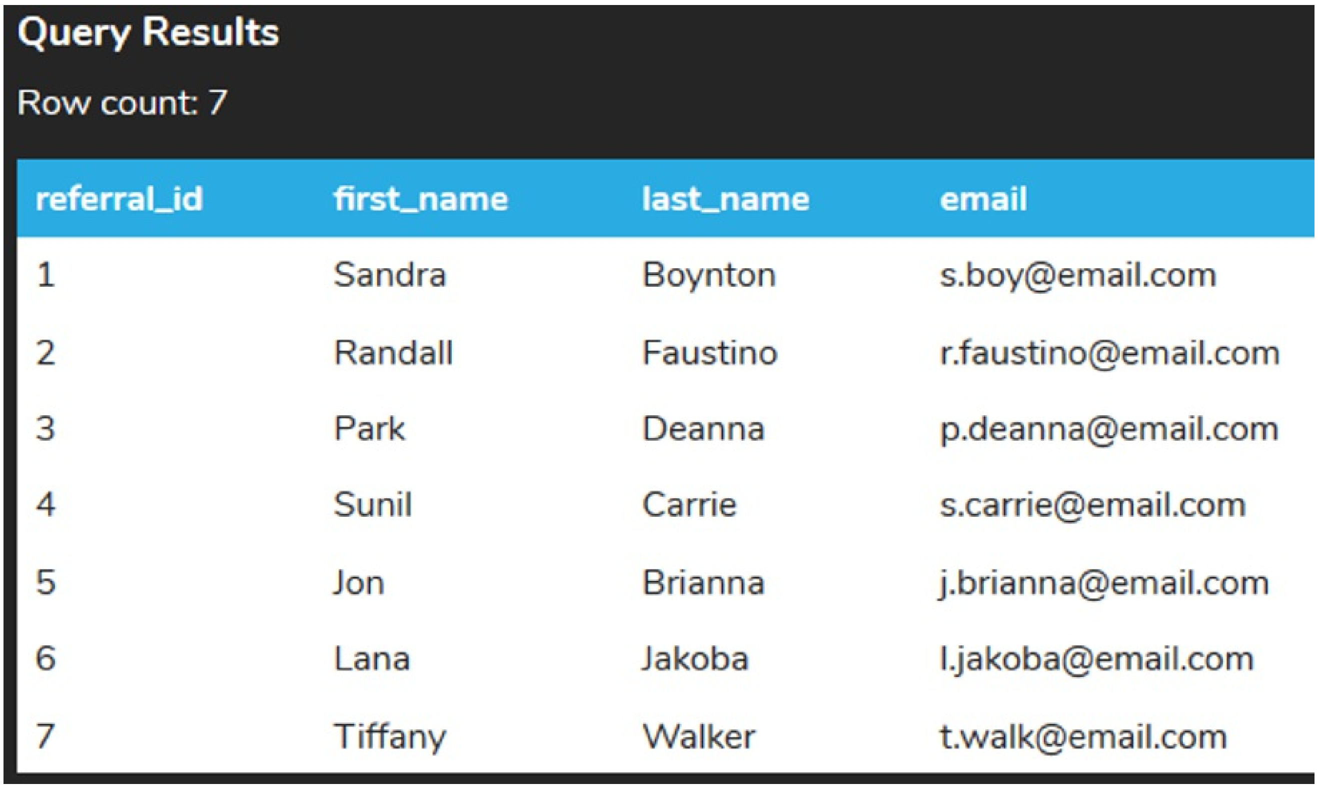
referral_id	first_name	last_name	email
1	Sandra	Boynton	s.boy@email.com
2	Randall	Faustino	s.boy@email.com
3	Park	Deanna	p.deanna@email.com
4	Sunil	Carrie	s.carrie@email.com
5	Jon	Brianna	j.brianna@email.com
6	Lana	Jakoba	l.jakoba@email.com
7	Tiffany	Walker	t.walk@email.com

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. 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;
```

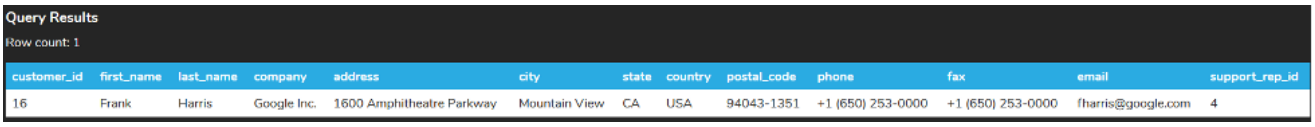

Query Results

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;
```

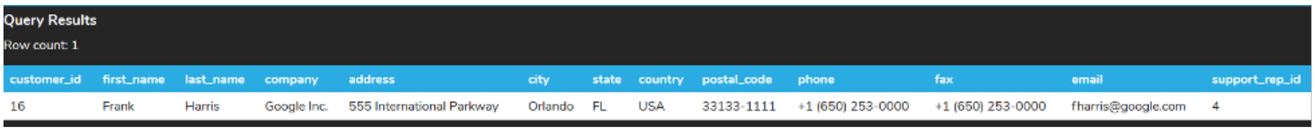

Query Results

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 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

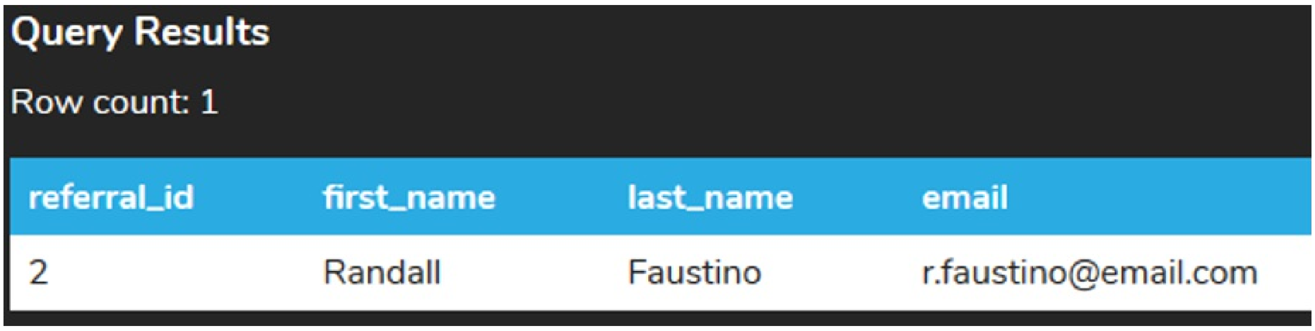
customer_id	first_name	last_name	company	address	city	state	country	postal_code	phone	fax	email	support_rep_id
16	Frank	Harris	Google Inc.	555 International Parkway	Orlando	FL	USA	33133-1111	+1 (650) 253-0000	+1 (650) 253-0000	fharris@google.com	4

2. 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 allows us to quickly validate the affected rows:



Query Results

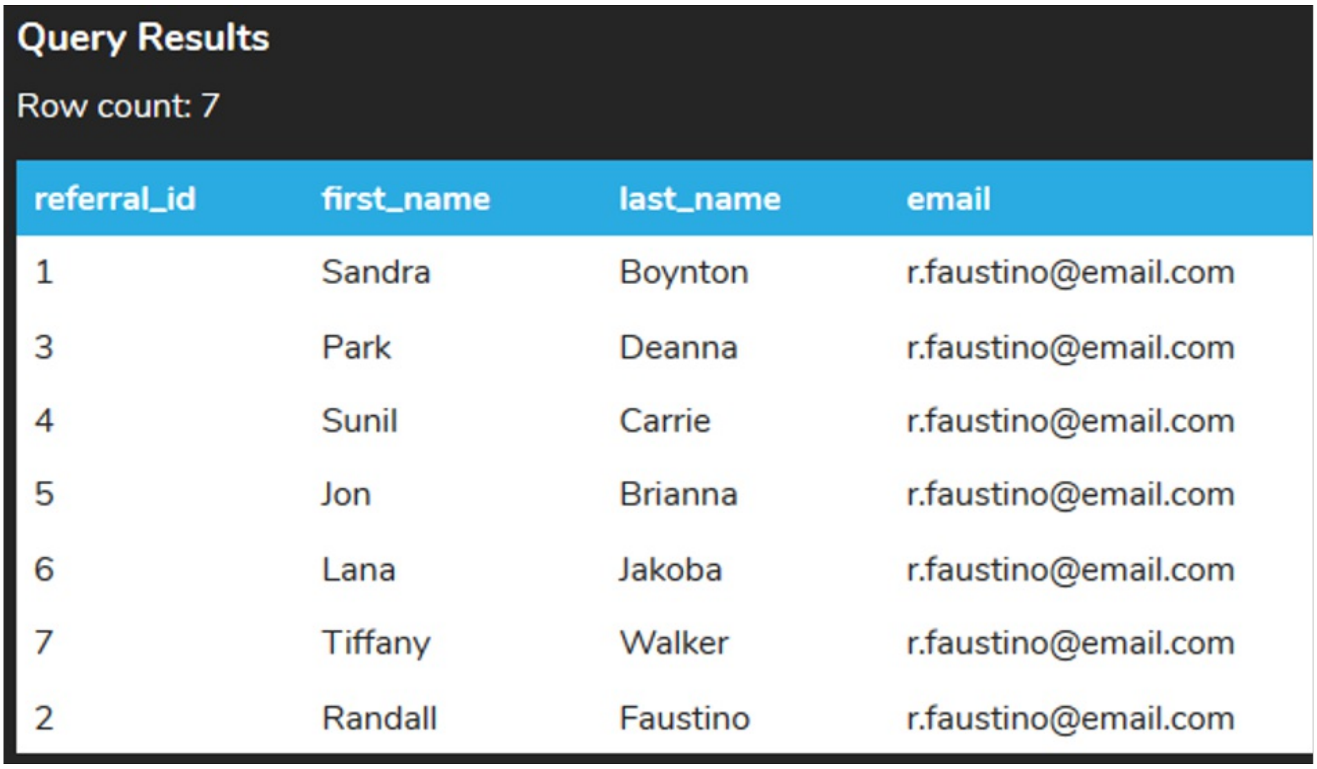
Row count: 1

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

Video Transcription

[MUSIC PLAYING] The update statement can be extremely useful to be able to make some modifications on the data itself. So if we have data that's already inserted to the table, we can make the updates and modify the data associated with them. The most common approach is to update utilizing the primary key as it ensures that we're going to update only that particular row.

The issue with modifying data based on other rows is that you could have repeating data, in which you might update rows that you don't want. For example, if we take a look at this table here, we have a contact ID with one, but the first name was Frank. If we scroll down a little bit further, we'll also see that contact ID nine has also Frank. So we're trying to modify utilizing the name Frank, that creates some issues in this case, being that it'll update both items.

Let's go ahead and make a modification, though, to the contact ID one, setting Frank's name to Fred instead. So the format of the statement will look like this. It will say update, and then the table name that we're going to set as a keyword, and then we identify which columns that we're going to modify. So in this case here, it's just going to be the first name. And then we'll send it to Fred in single quotes, and then we have the WHERE clause. The WHERE clause will work exactly the same as what you have in the select statement, so you can identify exactly what you want to set it to. So here we have contact ID equals to one.

We go ahead and update that. So if we go back and select from the table, we will be able to see that with contact ID equals 1, the name is updated to Fred. If we wanted to update multiple different columns at the same time, we can also do so by having multiple different items separated by commas within the set clause. So in this case here, if we change the first name to Fred, but also change last name, separate by a comma, to Flintstone. And we'll go ahead run that, and now we query that again, we'll be able to see that it's changed the first name and last name to Fred Flintstone.

[MUSIC PLAYING]

TRY IT

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.

SUMMARY

The UPDATE statement can be used to modify data in a single row.

Source: Authored by Vincent Tran