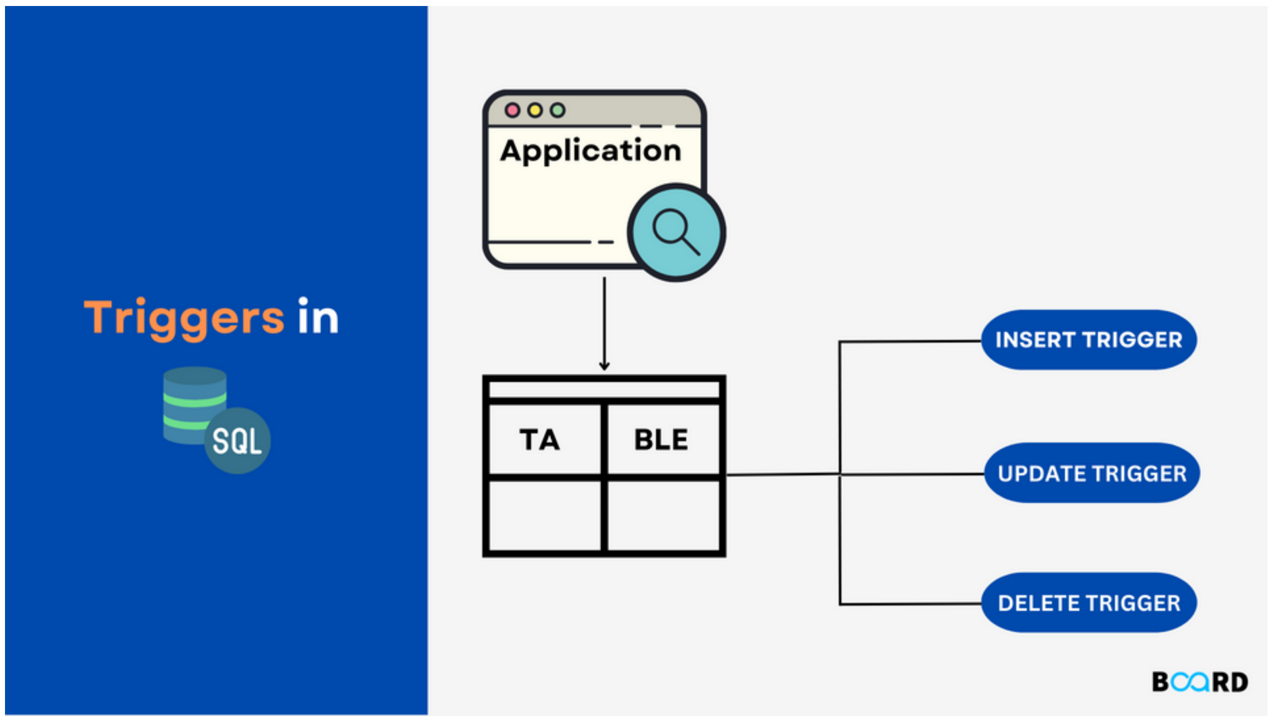
SQL
SERIES

How TO Create
SQL Triggers



Triggers are useful for automating repetitive tasks. You can just set up a trigger to do some calculation after every specific database action.

You can also set up a trigger to perform data validation tasks on a table.

A trigger gets fired when an INSERT, UPDATE or DELETE operation happens on a database table.

A trigger is fired per row, so if multiple rows of data are being inserted or deleted, each one still fires the action setup by the trigger. A trigger can be set to fire before or after an action.

In this post, you'll learn how to create triggers, how to drop them, and when they're useful.

How to Create A Trigger

To create a new trigger, use the CREATE TRIGGER command. This command has the following structure:

```
CREATE TRIGGER trigger_name
trigger_time
trigger_event
ON table_name
FOR EACH ROW
trigger_body
```

So here's a breakdown of every line:

- The keyword CREATE TRIGGER is mandatory and is followed by the name of the trigger. You'll use this name to refer to the trigger in future, and to delete the trigger if the need ever arises. This name should be unique per database.

- The trigger_time is a variable value that can only be either BEFORE or AFTER. This determines whether the trigger will fire before or after the event has happened.

- The trigger_event is another variable that has a limited number of possible options. This variable cannot be any value other than INSERT, UPDATE, or DELETE. It specifies what event to listen for.
- table_name is the name of the table the trigger should watch. This has to be the name of an existing table in your database, but it can be an empty table.
- The FOR EACH ROW is the other mandatory part of the trigger definition.
- trigger_body is the SQL query that you want to be run when this trigger is fired.

To create an example trigger, I will create a simple users table for practice.

```
CREATE TABLE
  users (
    fullname VARCHAR(120),
    email VARCHAR(120),
    username VARCHAR(30),
    password VARCHAR(60));
```

Now, we can create a simple trigger and attach it to this empty table. A trigger that encrypts string passwords before they are inserted using the MD5 function would make sense.

```
CREATE TRIGGER password_hasher BEFORE INSERT ON users FOR EACH ROW
SET
   NEW.password = MD5 (NEW.password);
```

This example is pretty straightforward and self-explanatory. But there's a NEW keyword there. This keyword gives you access to the new data being created and lets you use or modify the values as you like.

You can only modify these values if your set event_time is BEFORE. If the event_time is set to AFTER, the data has already been stored before getting to the trigger so it cannot be modified again.

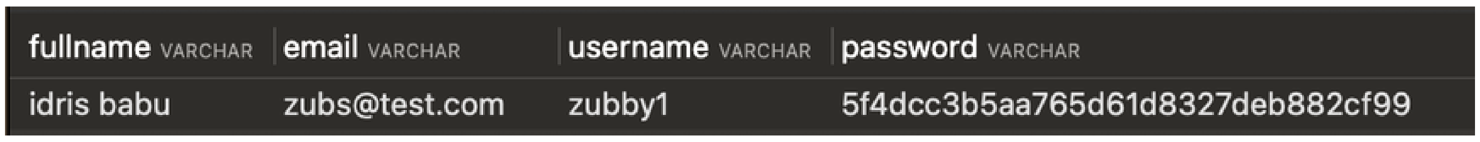
You can use the NEW keyword in INSERT and UPDATE events but not the DELETE event.

There's also the OLD keyword that you can use in DELETE and UPDATE event triggers that gives you access to the former values of the affected record. You can't use this keyword on an INSERT event because there is no previous record before new data is created.

To test this trigger, insert a row into the users table:

```
INSERT INTO
      users
VALUES
       (
          'idris babu',
          'zubs@test.com',
          'zubby1',
          'password'
      );
```

Check your table for the values. You should have something like this:



fullname VARCHAR	email VARCHAR	username VARCHAR	password VARCHAR
idris babu	zubs@test.com	zubby1	5f4dcc3b5aa765d61d8327deb882cf99

The password value has been correctly encrypted. 🥳

How to Drop a Trigger

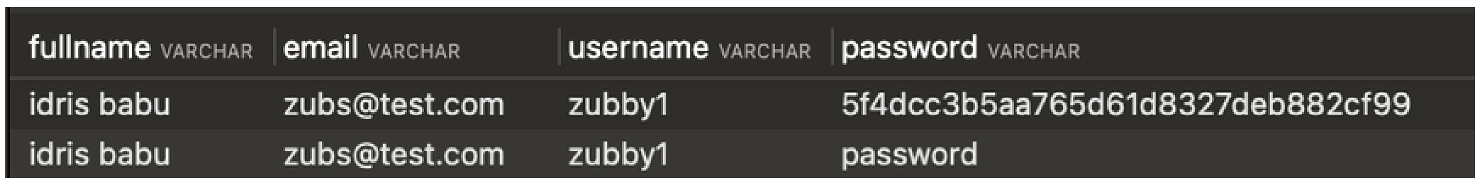
After creating a trigger, you might want to stop its execution for some reason. In this case, you can drop the trigger.

To drop the trigger, use the DROP TRIGGER command. The command only requires the name of the trigger. You can use the command like this:

```
DROP TRIGGER password_hasher;
```

Running this query will remove the trigger that we created above and every record inserted from now on will not have the password encrypted.

To test this, insert the same record as before, and check the result.



fullname VARCHAR	email VARCHAR	username VARCHAR	password VARCHAR
idris babu	zubs@test.com	zubby1	5f4dcc3b5aa765d61d8327deb882cf99
idris babu	zubs@test.com	zubby1	password

The newly created record doesn't have an encrypted password.

Something to keep in mind: if you drop the table completely, all associated triggers are also dropped automatically.

When to Use Triggers

- Logging: You can have a trigger to automatically write to another table on insertion, update, or deletion of record from a table.
- Data validation: You can write a trigger to ensure data is a certain type and correct values can be set when needed.
- Data syncronisation: You can use a trigger to keep related tables updated. For example, in an ecommerce table, every time a sales record gets created, a trigger can update the vendor's balance. Or if the vendor's record is deleted, a trigger can remove all their products.

I hope you now understand SQL triggers and when to use them so you can write better queries.

If you have any questions or suggestions, write in the comments.

HELPFUL? REPOST

linkedin.com/in/ileonjose