***TRIGGER***

CREATE TRIGGER trigger\_name [BEFORE|AFTER|INSTEAD OF] event\_name

ON table\_name

[

-- Trigger logic goes here....

];

*CREATE [ CONSTRAINT ] TRIGGER name { BEFORE | AFTER | INSTEAD OF } { event [ OR ... ] }*

*ON table*

*[ FROM referenced\_table\_name ]*

*[ FOR [ EACH ] { ROW | STATEMENT } ]*

*[ WHEN ( condition ) ]*

*EXECUTE PROCEDURE function\_name ( arguments )*

*where event can be one of:*

*INSERT*

*UPDATE [ OF column\_name [, ... ] ]*

*DELETE*

*TRUNCATE*

| **When** | **Event** | **Row-level** | **Statement-level** |
| --- | --- | --- | --- |
| BEFORE | INSERT/UPDATE/DELETE | Tables | Tables and views |
| TRUNCATE | — | Tables |
| AFTER | INSERT/UPDATE/DELETE | Tables | Tables and views |
| TRUNCATE | — | Tables |
| INSTEAD OF | INSERT/UPDATE/DELETE | Views | — |
| TRUNCATE | — | — |

* If multiple triggers of the same kind are defined for the same event, they will be fired in alphabetical order by name.
* Triggers that are specified to fire INSTEAD OF the trigger event must be marked FOR EACH ROW, and can only be defined on views.
* BEFORE and AFTER triggers on a view must be marked as FOR EACH STATEMENT.
* In addition, triggers may be defined to fire for TRUNCATE, though only FOR EACH STATEMENT.
* SELECT does not modify any rows so you cannot create SELECT triggers.
* To create a trigger on a table, the user must have the TRIGGER privilege on the table. The user must also have EXECUTE privilege on the trigger function.
* Use DROP TRIGGER to remove a trigger.

***EXAMPLE:***

CREATE TABLE COMPANY(

ID INT PRIMARY KEY NOT NULL,

NAME TEXT NOT NULL,

AGE INT NOT NULL,

ADDRESS CHAR(50),

SALARY REAL

);

CREATE TABLE AUDIT(

EMP\_ID INT NOT NULL,

ENTRY\_DATE TEXT NOT NULL

);

CREATE OR REPLACE FUNCTION auditlogfunc() RETURNS TRIGGER AS $example\_table$

BEGIN

INSERT INTO AUDIT(EMP\_ID, ENTRY\_DATE) VALUES (new.ID, current\_timestamp);

RETURN NEW;

END;

$example\_table$ LANGUAGE plpgsql;

CREATE TRIGGER example\_trigger AFTER INSERT ON COMPANY

FOR EACH ROW EXECUTE PROCEDURE auditlogfunc();

INSERT INTO COMPANY (ID,NAME,AGE,ADDRESS,SALARY)

VALUES (1, 'Paul', 32, 'California', 20000.00 );

Select \* from COMPANY;

Select \* from AUDIT;

SELECT \* FROM pg\_trigger;

DROP TRIGGER example\_trigger on COMPANY;

**WORK OUT QUERY:**

CREATE TRIGGER check\_update

BEFORE UPDATE ON accounts

FOR EACH ROW

EXECUTE PROCEDURE check\_account\_update();

CREATE TRIGGER check\_update

BEFORE UPDATE ON accounts

FOR EACH ROW

WHEN (OLD.balance IS DISTINCT FROM NEW.balance)

EXECUTE PROCEDURE check\_account\_update();

Uses for triggers:

* Enforce business rules
* Validate input data
* Generate a unique value for a newly-inserted row in a different file.
* Write to other files for audit trail purposes
* Query from other files for cross-referencing purposes
* Access system functions
* Replicate data to different files to achieve data consistency