Triggers

Triggers

Custom functions

Attached to tables

Executed when specific operation is performed e.g. INSERT, UPDATE...

"Event-driven"

Advanced skill level

Triggers

Executed before or after: INSERT, UPDATE, DELETE, or TRUNCATE

Executed once per modified row, or once per SQL statement:

When	Event	Row-level	Statement-level
BEFORE	INSERT/UPDATE/DELETE	✓	✓
BEFORE	TRUNCATE		✓
AFTER	INSERT/UPDATE/DELETE	✓	✓
AFTER	TRUNCATE		✓

For UPDATE events, may specify list of columns

Triggers: Syntax


```
CREATE TRIGGER trigger_name
{ BEFORE | AFTER }
{ INSERT | UPDATE [ OF column_name(s) ] | DELETE | TRUNCATE }
ON table_name
{ FOR EACH ROW | FOR EACH STATEMENT }
WHEN ( condition )
EXECUTE FUNCTION function_name ( arguments );
```

Triggers: Example


```
CREATE TRIGGER check_update
BEFORE UPDATE ON accounts
FOR EACH ROW
EXECUTE FUNCTION check_account_update();
```

Triggers: Example


```
CREATE TRIGGER check_update
BEFORE UPDATE OF balance ON accounts
FOR EACH ROW
EXECUTE FUNCTION check_account_update();
```

Triggers: Example


```
CREATE TRIGGER check_update
BEFORE UPDATE ON accounts
FOR EACH ROW
WHEN (OLD.balance != NEW.balance)
EXECUTE FUNCTION check_account_update();
```

Triggers: Example


```
CREATE FUNCTION function_name(arguments)
RETURNS trigger
AS 'function body text'
LANGUAGE plpgsql;
```

Function body written in SQL Procedural Language (PL/pgSQL)

Trigger functions


```
CREATE FUNCTION function_name(arguments)
RETURNS trigger
AS 'function body text'
LANGUAGE plpgsql;
```

```
CREATE FUNCTION function_name()
RETURNS trigger
AS $$
function body text
$$
LANGUAGE plpgsql;
```

Use dollar-quotes for convenience

Summary

Advanced feature for special use cases

Event-driven design

May execute BEFORE or AFTER an UPDATE, INSERT, DELETE, or TRUNCATE

May be executed once per statement, or once per modified row

Brings functional logic into database layer

Requires learning procedural language syntax