



# Databases

## PL/pgSQL – Triggers

Marco Vieira

**Bachelor in Informatics Engineering**  
*Department of Informatics Engineering*  
University of Coimbra  
2020/2021

2020/2021, Lesson #11 - TP

1



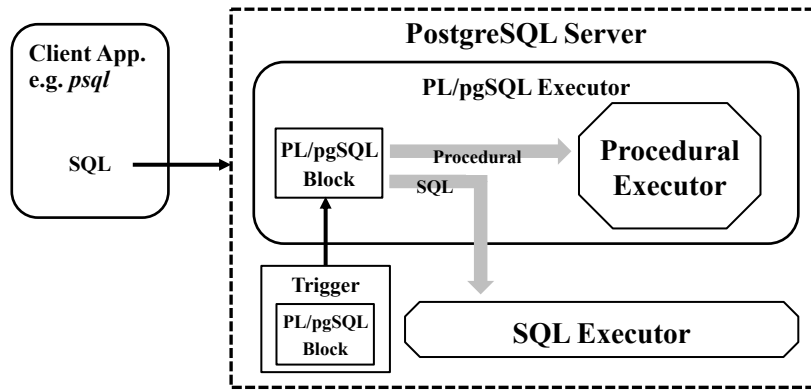
## Outline

---

- Triggers
- Why Triggers?
- Creating a Trigger
  - Activation event
  - Activation instant
  - Trigger type
  - Trigger restriction
- Trigger action

2

# Triggers

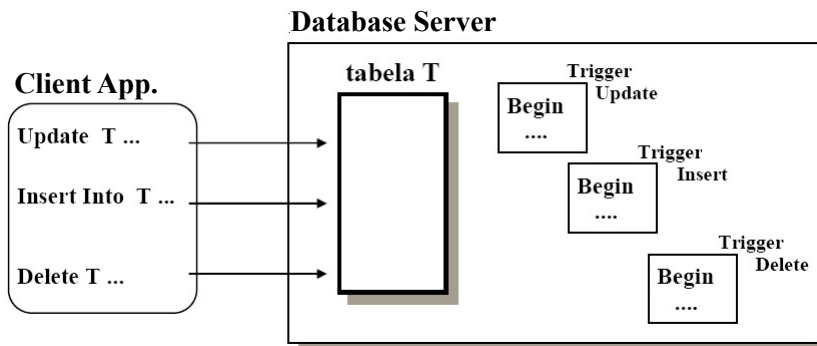


`insert into students values('António Silva', 23212, ...);`

3

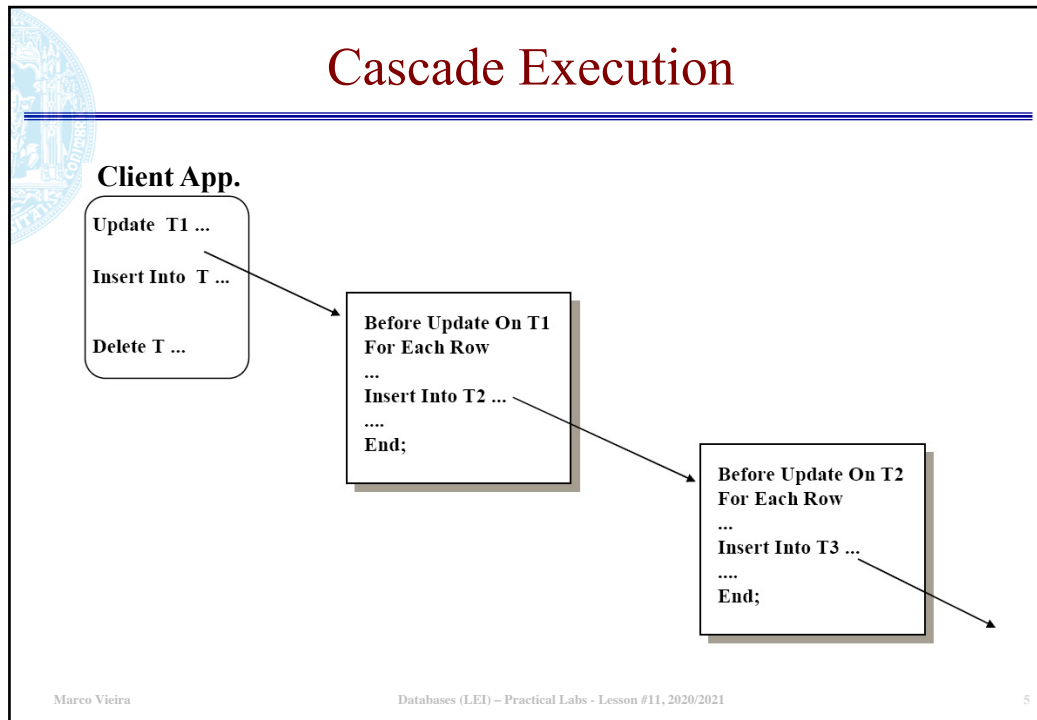
# Triggers: The Concept

- Program that runs automatically when some event occurs
  - e.g., insert, update and delete on table



4

## Cascade Execution




5

## Why Triggers? Examples...

- Implement advanced security mechanisms
- Implement complex integrity restrictions
- Allow the logging of all the operations performed
- Support auditing services (review of operations)
- Keep synchronous replicas
- Automatic computation of derived attributes
- Avoid invalid transactions
- Collect statistics
- ...

The slide features a circular seal on the left side. At the bottom, the text "Marco Vieira" is on the left, "Databases (LEI) – Practical Labs - Lesson #11, 2020/2021" is in the center, and a small "6" is on the right.

6




## Trigger Definitions

- Activation **event**
- Activation **instant**
- Trigger **type**
- Trigger **restriction**
- Trigger **action**

Marco Vieira Databases (LEI) – Practical Labs - Lesson #11, 2020/2021 7

7



## Creating a Trigger

- **Instant**: before, after, instead of
- **Event**: insert, update [of...], delete, truncate
- **Type**: for ach row, for each statement
- **Restriction**: when
- **Action**:
  - Function to be executed when the trigger is activated

```
create trigger name
{ before | after | instead of } { event [ or ... ] }
on table
[ for [ each ] { row | statement } ]
[ when ( condition ) ]
execute procedure proc_name ( arguments );
```

Marco Vieira Databases (LEI) – Practical Labs - Lesson #11, 2020/2021 8

8

## Activation Instant

- **BEFORE**
  - The trigger is executed immediately before the execution of the command that activates the trigger
- **AFTER**
  - The trigger is executed immediately after the execution of the command that activates the trigger
- It is possible to have one trigger being executed before and another one after

9

## Activation Event

- Event or SQL command SQL that activates the trigger to execute
- It can be an **INSERT, UPDATE or DELETE** in the table to which the trigger is associated
  - There are other event-based triggers (not seen in the course)
- **Several events can be combined** for the same trigger

```
create trigger t1
before update of sal on emp
...
```

```
create trigger t2
after insert on emp
...
```

```
create trigger t3
after insert or update or delete on emp
...
```

10



# Trigger Type

---

- **Row:**
  - Defined using the clause **FOR EACH ROW**
  - Activated for each line affected by the command that activates the trigger
    - e.g., if the trigger is activated by an update that changes 20 lines of a table, then the trigger will be executed 20 times (or less, depending on the restriction)
- **Statement:**
  - Defined using the clause **FOR EACH STATEMENT**
  - Executed only once, independently of the number of rows affected by the command that activates the trigger

11



# Instant/Type

---

- **before each statement**
  - Executed only once before the command that activates the trigger
- **before each row**
  - Executed before changing each line affected by the command that activates the trigger
- **after each statement**
  - Executed only once after the command that activates the trigger
- **after each row**
  - Executed after changing each line affected by the command that activates the trigger

12

## Trigger Restriction

- The trigger is executed only if the logical expression in the restriction (when) is true

```
create trigger t
after update of sal on emp
for each row
when (new.sal > old.sal)
...
```

13

## Trigger Action

- Function that should be executed when the trigger is activated
  - Such function **returns trigger**
- Row triggers have access to the **old and new values** of the line whose manipulation activated the trigger
  - **new.column** – new value (only for triggers activated by insert and update)
  - **old.column** – old value (only for triggers activated by update and delete)
- The event that activated the trigger can be identified using the variable **TG\_OP**

```
...
if (tg_op='DELETE') then
    insert into mytab values(old.nemp);
elsif (tg_op='INSERT') then
    insert into mytab values(new.nemp);
...
```

14

## Example

```
create or replace function func_trig1() returns trigger
language plpgsql
as $$
begin
    insert into mytab values(new.ndep);
    return new;
end;
$$;
```

```
create trigger trig1
after insert on dep
for each row
execute procedure func_trig1();
```

```
insert into dep values(50,'Sales','Coimbra');
```

```
drop trigger trig1() on dep;
```

15

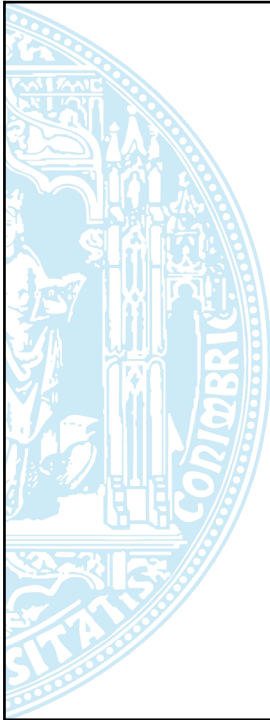
## Q&A



16



# Databases



## PL/pgSQL - Triggers

**Marco Vieira**

**Bachelor in Informatics Engineering**  
*Department of Informatics Engineering*  
University of Coimbra  
2020/2021

*2020/2021, Lesson #11 - TP*