

---

# Introduction aux triggers

---

**Nicolas Ribot - Licence GNU FDL - Version 1.1**

- ✦ Caractéristiques, fonctionnalités
- ✦ Variables
- ✦ Exemple



# Triggers: caractéristiques

- ❖ Trigger = déclencheur
- ❖ Automatiser des traitements sur une table ou vue en fonction de l'opération concernant la table:  
DELETE, INSERT, UPDATE, TRUNCATE
- ❖ Exemple: fabriquer un champ geometry de type POINT lors de l'insertion de valeurs X-Y provenant d'un GPS (cf. tp\_triggers\_simples.txt)
- ❖ Définition en deux étapes:
  - ❖ 1° Ecriture de la fonction du trigger, **sans arguments**
  - ❖ 2° Définition du trigger sur une table ou vue utilisant cette fonction
- ❖ Se déclenchent par ordre alphabétique si plusieurs triggers
- ❖ Peuvent être conditionnels (respect d'une condition avant déclenchement)

# Triggers: caractéristiques (suite)

- ❖ Déclenchement pour chaque ligne, ou pour chaque « statement »
- ❖ Déclenchement BEFORE, AFTER ou INSTEAD OF (pour les vues) opération
- ❖ Peut retourner NULL (l'opération est annulée) ou une ligne (record)
- ❖ Peut contenir des arguments (plusieurs triggers, une seule fonction)



# Triggers: variables

- ❖ NEW: *Record*, stocke la nouvelle valeur de la ligne
- ❖ OLD: *Record*, stocke l'ancienne valeur de la ligne
- ❖ TG\_NAME: nom du trigger courant
- ❖ TG\_WHEN: BEFORE ou AFTER ou INSTEAD OF
- ❖ TG\_LEVEL: niveau du trigger: ROW (row-level trigger) ou STATEMENT (stmt-level trigger)
- ❖ TG\_OP: quelle operation a déclenché le trigger INSERT, UPDATE, ou DELETE
- ❖ TG\_RELID: OID de la ligne concernée
- ❖ TG\_NARGS: le nombre d'arguments passés à la fonction du trigger.
- ❖ TG\_ARGS: *text[]*, le tableau des arguments passés à la fonction du trigger

# Triggers: exemple

```
CREATE TABLE xy (  
    id serial primary key,  
    x double precision not null,  
    y double precision not null,  
    geom geometry(Point, 4326)  
);
```

```
CREATE OR REPLACE FUNCTION xyToGeom()  
RETURNS trigger AS $xyToGeom$  
BEGIN  
    NEW.geom := st_setSRID(st_makePoint(NEW.x,  
    NEW.y), 4326);  
    RETURN NEW;  
END;  
$xyToGeom$ LANGUAGE plpgsql;
```

```
CREATE TRIGGER xyToGeom  
BEFORE INSERT OR UPDATE ON xy  
FOR EACH ROW EXECUTE PROCEDURE xyToGeom();
```

```
insert into xy(x, y) values (12.34, 45.56);
```

```
select id, x, y, st_asText(geom) from xy ;
```



# Triggers: exemple2

```
CREATE TABLE emp (  
    empname text,  
    salary integer,  
    last_date timestamp,  
    last_user text  
);
```

```
CREATE FUNCTION emp_stamp() RETURNS trigger  
AS $emp_stamp$  
BEGIN  
    -- Check that empname and salary are  
    given  
    IF NEW.empname IS NULL THEN  
        RAISE EXCEPTION 'empname cannot  
be null';  
    END IF;  
    IF NEW.salary IS NULL THEN  
        RAISE EXCEPTION '% cannot have  
null salary', NEW.empname;  
    END IF;
```

```
    -- Who works for us when she must pay  
    for it?  
    IF NEW.salary < 0 THEN  
        RAISE EXCEPTION '% cannot have a  
negative salary', NEW.empname;  
    END IF;
```

```
    -- Remember who changed the payroll  
    when  
    NEW.last_date := current_timestamp;  
    NEW.last_user := current_user;  
    RETURN NEW;  
END;
```

```
$emp_stamp$ LANGUAGE plpgsql;
```

```
CREATE TRIGGER emp_stamp BEFORE INSERT OR  
UPDATE ON emp  
FOR EACH ROW EXECUTE PROCEDURE emp_stamp  
();
```