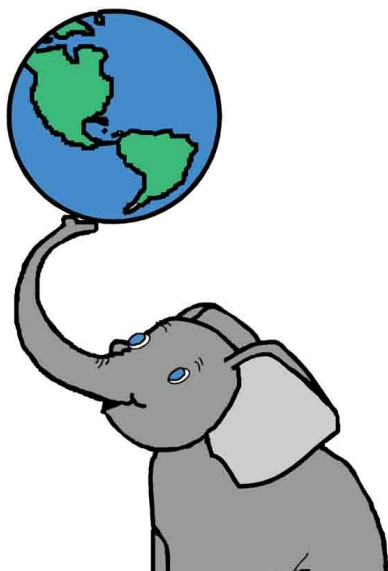


Exercice 12 et sa correction



version 1.3

Septembre 2019



Table des matières

I - Exercice : 12 - Vue modifiable	5
Solution des exercices	9

Exercice : 12 - Vue modifiable

Vous allez maintenant faire un exercice un peu plus compliqué (Vue modifiable avec l'utilisation de **INSTEAD OF**)

Question

[Solution n°1 p 9]

Nous souhaitons gérer le positionnement des étiquettes de la table *zonage* et pouvoir adopter une étiquette personnalisée différente du libellé dans une nouvelle table *zonage_etiq* spécifique à cette fonctionnalité (pour ne pas mélanger les données de présentation des étiquettes dans les données attributaires de la table *zonage*).

Cette nouvelle table aura pour attributs :

- gid : INTEGER (clef primaire)
- libelle_etiq : TEXT
- x_etiq : FLOAT
- y_etiq : FLOAT

Un zonage pourra avoir de 0 à 1 étiquette.

L'objectif est de proposer à l'utilisateur une vue que l'on appellera *vue_zonage* sur laquelle il pourra modifier le libellé et la position des étiquettes et à l'aide de trigger sur cette vue d'intercepter l'événement **UPDATE** pour faire en réalité les modifications dans la table *zonage_etiq*.

1) Créer la table *zonage_etiq* dans le schema *travail*.

zonage_etiq
<input type="checkbox"/> gid
<input type="checkbox"/> libelle_etiq
<input type="checkbox"/> x_etiq
<input type="checkbox"/> y_etiq

2) créer la vue *vue_zonage* par jointure avec les spécifications suivantes :

SI *zonage_etiq.gid* est NULL ALORS *vue_zonage.libelle_etiq* = *zonage.libelle* SINON *vue_zonage.libelle_etiq* = *zonage_etiq.libelle_etiq*

SI *zonage_etiq.gid* est NULL ALORS *vue_zonage.x_etiq* = « Coord X de centroide de *zonage* » SINON *vue_zonage.x_etiq* = *zonage_etiq.x_etiq*

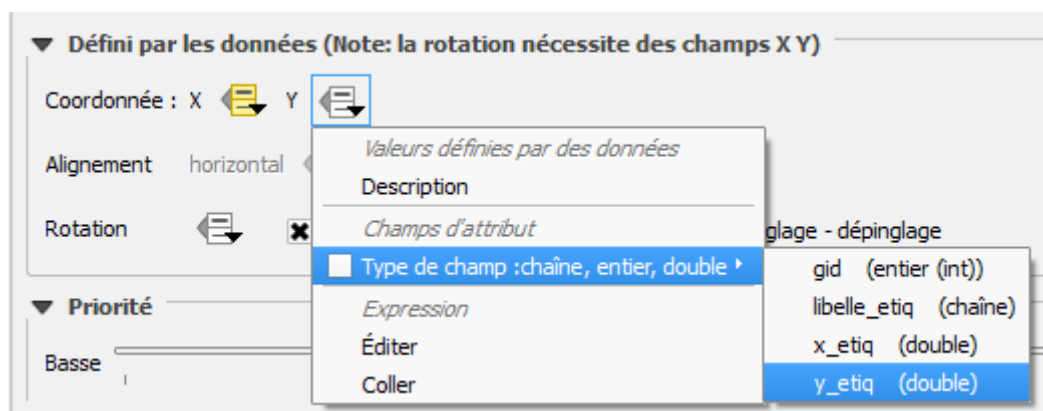
SI *zonage_etiq.gid* est NULL ALORS *vue_zonage.y_etiq* = « Coord Y de centroide de *zonage* » SINON *vue_zonage.y_etiq* = *zonage_etiq.y_etiq*



- 3) Créer une fonction `maj_zonage_etiq()` qui réalise l'algorithme suivant :
 SI OLD.gid « existait déjà dans la liste des » `zonage_etiq.gid` ALORS
 --> « mettre à jour » `zonage_etiq`
 -----> `zonage.libelle_etiq` avec le nouveau libelle en majuscule
 -----> `zonage.x_etiq` avec le nouveau `x_etiq`
 -----> `zonage.y_etiq` avec le nouveau `y_etiq`
 SINON
 --> « insérer » dans `zonage_etiq` un nouvel enregistrement (avec les nouvelles valeurs)
 - 4) créer le trigger sur la vue `vue_zonage`.
 - 5) faire un essai en modification d'étiquette en ajoutant la vue `vue_zonage` au canvas sous QGIS.
- vérifier les attributs de la table :

	gid	libelle_etiq	x_etiq	y_etiq
0	17	EXEMPLE DE LIBE...	199270.07085484	6274830.781156...
1	18	ALAIN	166298.2968326...	6267408.563627...

paramétrer l'emplacement des étiquettes dans les propriétés en indiquant les champs `x_etiq` et `y_etiq`



étiqueter la couche avec `libelle_etiq`

basculer en mode édition et personnaliser des étiquettes avec les boutons



enregistrer les modifications et vérifier le contenu de la table attributaire.

Envoyer vos différentes requêtes SQL aux tuteurs :

1. requête de création de la table zonage_etiq
2. requête de création de la vue vue_zonage
3. requête de création de la fonction maj_zonage_etiq
4. requête de création du trigger

Important : On ne cherche pas ici à pouvoir faire de la création de nouvelles entités directement dans *vue_zonage*. C'est pourquoi on n'intercepte pas l'événement **INSERT**.

La création de nouvelles entités doit toujours se faire dans la couche *zonage*.

Indices :

*Les spécifications de la jointure seront réalisées par utilisation de **CASE WHEN... THEN ... ELSE** dans la clause SELECT.*

*la fonction inclura un **IF...THEN... ELSE... END IF**. Le test portera sur le fait que OLD.gid (l'ancien gid avant modification) est ou non présent dans la liste des gid existants dans zonage_etiq.gid. Cette liste sera obtenue par **SELECT gid FROM travail.zonage_etiq**. Pour vérifier l'appartenance on utilisera donc le test.*

IF OLD.gid IN (SELECT gid FROM travail.zonage_etiq)

*Le trigger utilisera l'événement **INSTEAD OF UPDATE***



Solution des exercices

> Solution n°1 (exercice p. 5)

1) création de la table zonage_etiq

```
CREATE TABLE travail.zonage_etiq (gid SERIAL NOT NULL PRIMARY KEY,
libelle_etiq TEXT, x_etiq FLOAT, y_etiq FLOAT)
```

2) création de la vue vue_zonage :

```
CREATE OR REPLACE VIEW travail.vue_zonage AS
SELECT
zon.gid,
(CASE WHEN ze.gid IS NULL THEN zon.libelle ELSE ze.libelle_etiq END) AS
libelle_etiq,
(CASE WHEN ze.gid IS NULL THEN ST_X(ST_Centroid(geom)) ELSE ze.x_etiq
END) AS x_etiq,
(CASE WHEN ze.gid IS NULL THEN ST_Y(ST_Centroid(geom)) ELSE ze.y_etiq
END) AS y_etiq,
zon.geom
FROM travail.zonage AS zon
LEFT JOIN travail.zonage_etiq AS ze ON zon.gid=ze.gid;
```

3) création de la fonction :

```
CREATE FUNCTION travail.maj_zonage_etiq()
RETURNS trigger AS
$BODY$
BEGIN
IF OLD.gid IN (SELECT gid FROM travail.zonage_etiq) THEN
UPDATE travail.zonage_etiq
SET (libelle_etiq, x_etiq, y_etiq) = (UPPER(NEW.libelle_etiq),
NEW.x_etiq, NEW.y_etiq)
WHERE gid=OLD.gid;
ELSE
INSERT INTO travail.zonage_etiq
VALUES(NEW.gid, UPPER(NEW.libelle_etiq), NEW.x_etiq, NEW.y_etiq);
END IF;
RETURN NEW;
END;
$BODY$
LANGUAGE plpgsql;
```

4) création du trigger

```
CREATE TRIGGER maj_etiquette INSTEAD OF UPDATE
ON travail.vue_zonage
FOR EACH ROW
EXECUTE PROCEDURE travail.maj_zonage_etiq();
```

**Complément : Utilisation de la fonction coalesce()**

Dans les *fonctions conditionnelles*¹ il n'y a pas que le CASE WHEN...

On peut par exemple utiliser la fonction *COALESCE(valeur [, ...])* qui renvoie le premier de ses arguments qui n'est pas nul.

```
-- requête de création de la vue vue_zonage
create or replace view travail.vue_zonage as
select
  zonage.gid as gid
  ,coalesce(travail.zonage_etiq.libelle_etiq,  travail.zonage.libelle)  as
  libelle_etiq
  ,coalesce(travail.zonage_etiq.x_etiq,st_x(st_centroid(travail.zonage.geo
m))) as x_etiq
  ,coalesce(travail.zonage_etiq.y_etiq,st_y(st_centroid(travail.zonage.geo
m))) as y_etiq
  ,travail.zonage.geom
from travail.zonage
left join travail.zonage_etiq on zonage.gid = zonage_etiq.gid
;

-- requête de création de la fonction maj_zonage_etiq
CREATE OR REPLACE FUNCTION travail.maj_zonage_etiq()
RETURNS "trigger" AS
$BODY$
BEGIN
  if (new.gid in (select gid from travail.zonage_etiq)) then
    update travail.zonage_etiq set
      libelle_etiq          =          upper(coalesce(NEW.libelle_etiq,
travail.zonage_etiq.libelle_etiq)),
      x_etiq = coalesce(NEW.x_etiq, travail.zonage_etiq.x_etiq),
      y_etiq = coalesce(NEW.y_etiq, travail.zonage_etiq.y_etiq)
    where travail.zonage_etiq.gid = OLD.gid;
  else
    insert into travail.zonage_etiq (
      gid,
      libelle_etiq,
      x_etiq,
      y_etiq
    ) values (
      NEW.gid,
      upper(NEW.libelle_etiq),
      NEW.x_etiq,
      NEW.y_etiq
    );
  end if;
```

1 - <https://docs.postgresql.fr/9.5/functions-conditional.html>

```
end if;
-- En option (non explicitement demandé) on peut traiter le cas de la
mise à jour de la géographie dans la vue_zonage
update travail.zonage
set geom = coalesce(NEW.geom, travail.zonage.geom)
where travail.zonage.gid = OLD.gid;
return new;
END;
$BODY$
LANGUAGE plpgsql;
```



Complément

On trouvera des exemples d'utilisation de triggers dans la documentation de PostgreSQL. Pour créer des tables d'audit on pourra également voir *cet exemple*² ou *encore*³ (en anglais) ou un article plus complet sur *l'historisation de l'évolution des données*.⁴

2 - https://wiki.postgresql.org/wiki/Audit_trigger

3 - http://workshops.boundlessgeo.com/postgis-intro/history_tracking.html

4 - http://blog.developpez.com/sqlpro/p8453/langage-sql-norme/historisation_de_l_evolution_des_donnees