Bases de données avancées
TD n° 3 (PL/pgSQL) : Déclencheurs
J. Darmont (https://eric.univ-lyon2.fr/jdarmont/), 22/07/25

Aide-mémoire : Définition de déclencheur

```
CREATE FUNCTION nomFonction() RETURNS TRIGGER AS $$
     [DECLARE]
          -- Déclarations
     BEGIN
          -- Instructions
     END
$$ LANGUAGE plpgsql;

CREATE TRIGGER nomDeclencheur
     BEFORE | AFTER
     INSERT | DELETE | UPDATE | [INSERT] [[OR] DELETE] [[OR] UPDATE]
     ON nomTable
     FOR EACH ROW | FOR EACH STATEMENT
     EXECUTE PROCEDURE nomFonction();
```

Exercice 1 : Transformation automatique

1. Télécharger et exécuter le script SQL indiqué à l'adresse suivante, qui crée la table DEMO_STATES.

   https://eric.univ-lyon2.fr/jdarmont/docs/pgDemoStates.sql

Conformément aux données existantes, on souhaite que toutes les nouvelles données insérées dans la table soient en majuscules.

2. Définir un déclencheur avant insertion ou modification qui transforme systématiquement en majuscules les valeurs de ST et STATE_NAME, quel que soit leur format d'origine (utiliser la fonction UPPER).

3. Tester l'insertion et la modification de quelques n-uplets et vérifier le contenu de la table DEMO_STATES.

Exercice 2 : Contraintes d'intégrité conditionnelles

1. Télécharger et exécuter le script SQL indiqué à l'adresse suivante, qui crée la table EventRequest.

   https://eric.univ-lyon2.fr/jdarmont/docs/pgEventRequest.sql

2. Écrire un déclencheur, ainsi que la fonction associée, de nom checkDates( ) permettant de vérifier que, pour chaque nouvelle demande ou modification de demande dans la table EventRequest, la date d'autorisation dateAuth est renseignée. Si ce n'est pas le cas, interrompre l'exécution avec le message « Erreur d'autorisation ». Sinon, vérifier que la dateAuth est supérieure ou égale à la dateReq. Si ce n'est pas le cas, interrompre l'exécution avec le message « Erreur de date ».

3. Tester l'insertion ou la modification de cas d'erreur et un cas normal, puis vérifier le contenu de la table EventRequest.

Exercice 3 : Contrainte d'intégrité dynamique

Soit la table CLIBANQUE(idCli, nomCli, idConjoint#), où l'attribut idConjoint est l'identifiant (idCli) du conjoint du client courant. Il s'agit de coder un déclencheur avant insertion ou modification qui contrôle que le nom du/de la conjoint·e d'un·e client·e soit le même que celui de ce·tte client·e (plutôt conservateur, comme politique, n'est-ce pas ?). Si les deux noms diffèrent, une exception doit interrompre l'insertion ou la modification des données.

1. Télécharger et exécuter le script SQL indiqué à l'adresse suivante, qui crée la table CLIBANQUE.

https://eric.univ-lyon2.fr/jdarmont/docs/pgCliBanque.sql

2. Créer le déclencheur et la fonction associée.

3. Insérer et modifier quelques n-uplets dans la table CLIBANQUE. Conclusion ?

Exercice 4 : Statistiques d'utilisation

Il s'agit de recenser des statistiques de mise à jour des données (insertions, modifications, suppressions) de la table EMP.

1. Soit la table STATS (typeMaj, nbMaj, timestampModif), à créer à l'aide du script SQL ci-dessous.

https://eric.univ-lyon2.fr/jdarmont/docs/pgSTATS.sql

STATS

typeMaj	nbMaj	timestampModif
INSERT	0	NULL
UPDATE	0	NULL
DELETE	0	NULL

2. Définir un déclencheur après insertion, modification ou suppression dans la table EMP qui met à jour automatiquement la table STATS. Tester son fonctionnement en effectuant quelques modifications dans la table EMP.

Indications :

- Le type de mise à jour est donné par la variable système `TG_OP`.
- Le timestamp courant est donné par la fonction `NOW()`.

3. Tester l'effet des clauses FOR EACH ROW et FOR EACH STATEMENT sur le comportement du déclencheur en utilisant une requête qui modifie plusieurs n-uplets (ex. `UPDATE emp SET sal = sal * 1.05`).

Exercice 5 : Rafraîchissement de vue matérialisée

1. Créer la table DEMO_CUSTOMERS depuis le script SQL ci-dessous.

https://eric.univ-lyon2.fr/jdarmont/docs/pgDemoCustomers.sql

2. Créer une table nommée CUSTNAMES à partir de la requête SQL suivante.

```
CREATE TABLE CUSTNAMES AS
SELECT CUSTOMER_ID, CUST_FIRST_NAME, CUST_LAST_NAME FROM DEMO_CUSTOMERS;
```

3. Définir un déclencheur après insertion, modification ou suppression dans la table DEMO_CUSTOMERS qui répercute toutes les mises à jour de DEMO_CUSTOMERS dans CUSTNAMES.

4. Tester le fonctionnement du déclencheur en insérant, modifiant puis supprimant un n-uplet dans la table DEMO_CUSTOMERS. Vérifier à chaque étape si la mise à jour est bien répercutée dans CUSTNAMES.