M1 Informatique – Année 2024-2025
Bases de données avancées
TD n° 1 (PL/pgSQL) : Requêtes simples, curseurs implicites
J. Darmont (https://eric.univ-lyon2.fr/jdarmont/), 11/09/24

Aide-mémoire PL/pgSQL

Bloc PL/SQL	`DECLARE`
	`  -- Déclarations : variables, constantes, tableaux`
	`  -- curseurs, fonctions`
	`BEGIN`
	`  -- Instructions du programme principal`
	`EXCEPTION`
	`  -- Traitement des erreurs à l'exécution`
	`END`

Déclaration de variable	`nom_variable TYPE_VARIABLE;`
Affectation	`nom_variable := valeur;` `SELECT attribut INTO nom_variable FROM table;`

Tests	`IF condition1 THEN`
	`  -- Instructions`
	`ELSIF condition2 THEN  -- (Optionnel)`
	`  -- Instructions`
	`ELSE  -- (Optionnel)`
	`  -- Instructions`
	`END IF;`
	`CASE nom_variable`
	`  WHEN valeur1 THEN -- Instructions`
	`  WHEN valeur2 THEN -- Instructions`
	`  ELSE  -- Instructions`
	`END CASE;`

Boucles	`FOR iterateur IN [REVERSE] min..max [BY pas] LOOP`
	`  -- Instructions`
	`END LOOP;`
	`WHILE condition LOOP`
	`  -- Instructions`
	`END LOOP;`
	`LOOP -- Répéter jusqu'à`
	`  -- Instructions`
	`  EXIT WHEN condition`
	`END LOOP;`

Curseur implicite (non lié)	`FOR nuplet IN requete LOOP -- Type de nuplet : RECORD`
	`  -- Instructions`
	`  -- Ex. nomVariable := nuplet.attribut;`
	`END LOOP;`

Exceptions - *BEGIN*	`RAISE EXCEPTION 'message d''erreur';`
- *EXCEPTION*	`WHEN codeException THEN` `  -- Instructions;`
Changement de mot de passe	`ALTER USER nom_login WITH PASSWORD 'nouveau_mdp';`
Documentation PL/pgSQL	https://docs.postgresql.fr/17/plpgsql.html

Logiciel client : DBeaver Community

- Création d'une nouvelle connexion : icône .
  - Sélectionner PostgreSQL.
  - Renseigner le nom du serveur hôte, la base de données, le nom d'utilisateur et le mot de passe fournis par l'enseignant·e.
  - Tester la connexion puis terminer. Double-cliquer sur la connexion pour l'activer.
- Création d'un éditeur SQL : icône ou **F3**.
- Exécution d'une requête SQL : Ctrl + Entrée – d'un script PL/pgSQL : **Alt + X**.

Exercice 1

1. Télécharger le script SQL indiqué à l'adresse suivante, qui crée la table PARTS (pièces). L'ouvrir dans DBeaver, copier/coller le code dans l'éditeur SQL, puis l'exécuter.

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

2. Écrire une fonction PL/pgSQL nommée nbParts() qui retourne un nombre entier. Y définir une variable entière n, puis lui affecter le nombre total de pièces enregistrées dans la table PARTS. Retourner n.

3. Tester l'exécution de nbParts().

4. Dans le cas où le nombre de pièces est supérieur à 6, déclencher une exception qui interrompt la fonction (message d'erreur au choix). Tester.

5. Indiquer dans le message de l'exception le nombre de pièces. Tester.

Exercice 2

Soit la table EMP qui stocke des informations sur les employés d'une entreprise. On souhaite déterminer la proportion de managers parmi eux.

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

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

2. Écrire une fonction PL/pgSQL nommée propMgr(), retournant un nombre réel et permettant de :

- compter le nombre total de n-uplets dans la table EMP et stocker le résultat dans une variable ;
- compter le nombre d'employés dont la fonction (JOB) est MANAGER dans la table EMP et stocker le résultat dans une deuxième variable ;
- calculer la proportion (en pourcentage), stocker le résultat dans une troisième variable et retourner le résultat.

3. Tester l'exécution de propMgr().

4. Inclure dans le programme précédent un traitement d'exception permettant à la fonction propMgr() de lever un *warning* et de renvoyer la valeur NULL si la table EMP est vide (ce qui provoque une division par zéro – détourner l'exception système division_by_zero). Tester le bon fonctionnement de l'exception en suivant la procédure suivante :

1. effacer le contenu de la table EMP (`DELETE FROM emp`) ;
2. exécuter la fonction propMgr() ;
3. réexécuter la partie du script pgEMP.sql qui insère les données.

5. Le warning s'affiche-t-il à l'écran ? Où ?

Exercice 3

1. Écrire une fonction PL/pgSQL nommée cat() qui retourne le nom de toutes vos tables. Pour cela, parcourir la requête `SELECT tablename FROM pg_tables WHERE tableowner = 'votreLogin'` à l'aide d'un curseur implicite.

2. Tester l'exécution de cat().

Exercice 4

1. Écrire une fonction PL/pgSQL nommée aug1000(), qui prend en paramètre un numéro de département (INT) et ne retourne rien (VOID). Cette fonction (qui est une procédure, en fait) a pour objet d'augmenter de 1000 $ le salaire (SAL) des employés de la table EMP qui gagnent déjà plus de 1500 $ et appartiennent au département (DEPTNO) dont le numéro a été passé en paramètre.

2. Consulter la table EMP, exécuter la fonction aug1000() pour le département 20 et vérifier que les mises à jour du salaire ont bien été effectuées.

3. Ajouter une exception à la fonction aug1000() : si le nouveau salaire d'un employé devient supérieur ou égal au plus haut salaire dans l'entreprise, il faut interrompre le traitement. Indiquer le nom (ENAME) de l'employé qui provoque l'erreur.

4. Exécuter la fonction aug1000() pour le département 20 jusqu'à ce que l'exception se produise. Est-ce que la dernière augmentation a eu lieu ?

Correction

```
-- Exercice 1

CREATE OR REPLACE FUNCTION nbParts() RETURNS INTEGER AS $$
     DECLARE
          n INTEGER;
     BEGIN
          SELECT COUNT(*) INTO n FROM parts;
          IF n > 6 THEN
               RAISE EXCEPTION 'Trop de pièces (%) !', n;
          END IF;
          RETURN n;
     END
$$ LANGUAGE plpgsql;

SELECT nbParts();


-- Exercice 2

CREATE OR REPLACE FUNCTION propMgr() RETURNS REAL AS $$
     DECLARE
          nTot INTEGER; -- Nombre total d'employés
          nMan INTEGER; -- Nombre de managers
     BEGIN
          SELECT COUNT(*) INTO nTot FROM emp;
          SELECT COUNT(*) INTO nMan FROM emp WHERE job = 'MANAGER';
          RETURN 100 * nMan / nTot ::REAL;
     EXCEPTION
          WHEN division_by_zero THEN
               RAISE WARNING 'Division par zéro – pas d''employé';
               RETURN NULL;
     END
$$ LANGUAGE plpgsql;

SELECT propMgr();

-- Le warning est transmis à DBeaver et s’affiche dans la fenêtre Sortie.


-- Exercice 3

CREATE OR REPLACE FUNCTION cat() RETURNS SETOF VARCHAR AS $$
     DECLARE
          nuplet RECORD;
     BEGIN
          FOR nuplet IN
               SELECT tablename FROM pg_tables WHERE tableowner = 'darmont'
          LOOP
               RETURN NEXT nuplet.tablename;
          END LOOP;
          RETURN;
     END
$$ LANGUAGE plpgsql;

SELECT * from cat();
```

-- Exercice 4

```
CREATE OR REPLACE FUNCTION aug1000(noDept INT) RETURNS VOID AS $$
     DECLARE
          nuplet emp%ROWTYPE;
          salMax emp.sal%TYPE;
          nouvSal emp.sal%TYPE;
     BEGIN
          SELECT MAX(sal) INTO salMAX FROM emp;
          FOR nuplet IN
               SELECT * FROM emp WHERE deptno = NoDept AND sal > 1500
          LOOP
               nouvSal := nuplet.sal + 1000;
               IF nouvSal >= salMax THEN
                   RAISE EXCEPTION '% ne peut pas gagner plus que le président!',
                         nuplet.ename;
               END IF;
               UPDATE emp SET sal = nouvSal WHERE empno = nuplet.empno;
          END LOOP;
     END
$$ LANGUAGE plpgsql;

SELECT aug1000(20);
```