***COURS BD***

* DML : Data Manipulation Language 🡪 INSERT, UPDATE pour les données (possibilité rollback/commit)
* DDL : Data Definition Language 🡪 CREATE, ALTER, DROP pour structure de donnée(tables…) **(pas de possibilité de rollback, commit)**
* DCL : Data Control Language 🡪 GRANT, REVOKE
* TCL : Transaction Control Language 🡪 ROLLBACK, COMMIT

Langage 4G : PL/SQL  (oracle), Transac SQL (sql serveur)

Le PL/SQL sert à faire :

* des blocs anonymes (écrits,exe,suppr)
* des blocs nommés (objets au même titre que des tables) **Procédures, fonctions**
* Triggers (morceaux de code générés auto lors d’un évennement)

camelCase/PascalCase

www.ggevrin.com/enseignement/lyon1/[S2105C/](http://www.ggevrin.com/enseignement/lyon1/S2105C/)

**STRUCTURE DU LANGAGE :**

Set serveroutput on

**DECLARE**

-- déclarations de variables

{ v\_nom *(v\_ peut varier)* VARCHAHR2(20) ;

v\_sal NUMBER(5,2) ; (5 chiffres dont 2 décimales) }

* --> c’est mieux d’écrire que cette variable est du même type que du champ sal de la table emp, si v\_sal devient de type NUMBERS(7,2) ça devient le bordel

v\_nom emp.ename%TYPE ;

v\_sal emp.sal%TYPE ;

v\_emp%ROWTYPE ; ( variable composite un peu comme une struct en c )

**BEGIN**

-- instructions ;

V\_nom := ‘&v\_nom’ ; 🡪(sert à entrer manuellement v\_nom) ‘&v\_nom’ quote pour chaine

SELECT ename, sal INTO v\_nom, v\_sal (premier dans premier, deuxième dans deuxième)

FROM emp

WHERE empno = 18 ; (ces trois lignes ne marchent que si y’a qu’un résultat)

(si on veut afficher des trucs, le logiciel n’est pas conçu pour il faut donc rajouter le vert)

DBMS\_OUTPUT.PUT\_LINE(‘L employe’ || v\_nom ||’gagne’ || ‘v\_sal’) ;

V\_Nom : = UPPER(V\_nom) ; (pas oublier l’opérateur d’affectation « := »)

UPDATE emp

SET sal=v\_sal

WHERE matr=18 ;

COMMIT ;

SELECT \* INTO v\_emp

FROM emp

WHERE matr=42 ;

V\_emp.matr=666 ; (on lui donne un nouveau matricule)

INSERT INTO emp

VALUES v\_emp ;

COMMIT ;

**EXCEPTIONS**

--Traitement exceptions

**END** ;

**Les structures conditionnelles :**

**1 –**

**IF** conditions **THEN**

--instructions ;

**ELSE**

--instructions ;

**END IF ;**

**2 -**

**CASE** expression

**WHEN** valeur1 **THEN**

--instructions ;

**WHEN** valeur2 **THEN**

--instructions ;

**ELSE**

--instructions ;

**END CASE ;**

**3 –**

**LOOP**

**EXIT [WHEN** condition**]** (faut pas écrire les crochets, ça veut juste dire que c’est optionnel)

--instructions ;

**END LOOP ;**

**4 –**

**WHILE** conditions **LOOP**

--instructions ;

**END LOOP ;**

**5-**

**FOR** compteur **IN [REVERSE] inf..sup LOOP**

--instructions ;

**END LOOP ;**

**CURSEURS :**

**Attributs :**

* **%ROWCOUNT**
* **%FOUND**
* **%NOTFOUND**
* **%ISOPEN**

*Ex :*

DECLARE

v\_nb NUMBER ;

BEGIN

DELETE FROM emp

WHERE job=’SALESMAN’ ;

DBMS\_OUTPUT.PUTLINE(SQL%ROWCOUNT || ‘vendeurs desintégrés’) ;

END ;

* Implicites 1
* Explicites 2 : Déclarer

OPEN

FETCH

CLOSE

**DECLARE**

CURSOR emp IS

SELECT \* FROM emp WHERE deptno=10 ;

V\_emp c\_emp%ROWTYPE ;

**BEGIN**

OPEN c\_emp ; (éxécute l’odre qui est associé au curseur)

**LOOP**

FETCH c\_emp INTO v\_emp ; (dans une boucle, ça déplace le pointeur au record suiv et déplace dans v\_emp)

EXIT WHEN c\_emp %NOTFOUND ;

DBMS\_OUTPUT.PUT\_LINE (v\_emp.ename) ;

**END LOOP ;**

CLOSE c\_emp ; (libère la mémoire)

**END ;**

**Exceptions :**

* Déclaration 1
* Levée 2
* Récupération 3

***DECLARE***

1. ensalTropBas EXCEPTION ; (nom, type)

v\_sal emp.sal%TYPE ;

***BEGIN***

SELECT sal into v\_sal

FROM emp

WHERE ename =’GEVRIN’ ;

IF (v\_sal < 50 000 000 000) THEN

2) RAISE ensalTropBas ; (levée d’exception)

END IF ;

---------- AUTRES instructions

***EXCEPTION***

3) WHEN NO\_DATA\_FOUND THEN (exception standard SQL)

DBMS\_OUTPUT.PUT\_LINE (‘déjà viré’) ;

WHEN e\_salTropBas THEN

RAISE\_APPLICATION\_ERROR (-20001, ‘Augmentre-moi’ !) ; ( >20 000 param, et texte)

WHEN OTHERS THEN

DBMS\_OUTPUT.PUT\_LINE (SQLERRM) ; --SQL CODE

**DECLARE**

CURSOR c\_bidule (p\_deptno EMP.DEPTNO%TYPE) is

SELECT \* FROM emp WHERE deptno=p\_deptno;

~~V\_bidule c\_bidule %ROWTYPE ;~~

**BEGIN** (v\_bidule ROWTYPE %)

~~OPENc\_bidule ;~~ FOR v\_bidule IN c\_bidule (10) LOOP

~~L~~**~~OOP~~**

~~FETCH c\_bidule INTO v\_bidule ;~~

~~EXIT WHEN c\_bidule %NOTFOUND ;~~

DBMS\_OUTPUT.PUT\_LINE (v\_bidule.ename) ;

**END LOOP ;**

CLOSE c\_bidule ;

**END ;**