BASES DE DONNEES

PL/SQL

Résumé de la syntaxe

[DECLARE
Déclaration des variables **]**
BEGIN
Corps du programme
[EXCEPTION
Gestion des erreurs **]**
END

- **DECLARATIONS**
  - **Variables et constantes**

Types de données : **NUMERIC([,]), CHAR(), VARCHAR(), DATE, BOOLEAN , LONG** …
avec 2 types composés : **RECORD** , TABLE
et des correspondances : **%TYPE** , %ROWTYPE

nomVar [CONSTANT] {type | rel.att%TYPE | rel%ROWTYPE**} [{:= | DEFAULT} expression]**

  - **Déclaration des curseurs**

CURSOR nomCurs IS SELECT …

  - **Déclaration des exceptions**

nomExc EXCEPTION ;

- **AFFECTATION : =**
- **COMPARAISON = <> < > <= >=**
  IS NULL , IN , BETWEEN , LIKE , AND , OR , NOT

- **CONTROLE**
  - **Traitements conditionnels**

```
IF condition THEN
  …
[ ELSIF condition THEN
          ...              ]
[ ELSE
  …                ]
END IF ;
```

  - **Traitements itératifs**

```
LOOP
      …
   EXIT [WHEN condition]
      …
END LOOP ;
```

```
WHILE condition
LOOP
      ...
END LOOP ;
```

```
FOR compteur IN [REVERSE]
          born_inf .. born_sup
LOOP
      ...
END LOOP ;
```

- **EXCEPTIONS**
  - **Exceptions prédéfinies**

NO_DATA_FOUND , TOO_MANY_ROWS , DUP_VAL_ON_INDEX …

  - **Exceptions propres à l'application, créées par le programmeur**

Définies dans la section **DECLARE**.
Déclenchées par l'instruction **RAISE nomExc ;**

  - **Traitement des exceptions**

```
EXCEPTION
  WHEN nomExc1 THEN
          ...
  WHEN nomExcN THEN
          ...
  WHEN OTHERS THEN
          ...
END;
```

- **ERREURS**

RAISE_APPLICATION_ERROR(NumErr, Message)
avec -20000 < NumErr < -20999

Arrête l'exécution du code PL/SQL et renvoie un code d'erreur.

- **CURSEURS**

  - **Déclaration** *(cf bloc DECLARE)* — **CURSOR nomCurs IS SELECT … (SQL)**
  - **Ouverture** — **OPEN nomCurs**
  - **Défilement** — **FETCH nomCurs INTO var_1 [, ..., var_N]**
  - **Fermeture** — **CLOSE nomCurs**
  - **Attributs spécifiques** — **nomCurs%NOTFOUND** **nomCurs%FOUND** **nomCurs%ROWCOUNT** **nomCurs %ISOPEN**

- **REQUETES SQL**

  Le langage SQL est disponible mais

  Avec une nécessaire affectation dans le SELECT

  - **SELECT ... INTO listedevariables**
    FROM ...
    WHERE … [FOR UPDATE]

  Et une référence possible au n-uplet en cours lors de mises à jours ou supression :

  - **UPDATE / DELETE ...**
    WHERE … | [CURRENT OF nomcur]

- **PROCEDURES STOCKEES**

CREATE PROCEDURE nomProc [(arg_1 [.., arg_n)]] IS
[Déclarations locales]
BEGIN
...
... corps de la procédure
...
[EXCEPTION
... gestion des exceptions]
END nomProc ;

Avec arg_i [IN / OUT / IN OUT] type [{ := | DEFAULT } expression]

- ***Utilisation***
  *EXEC **nomProc** (…)*

- **FONCTIONS**

CREATE FUNCTION nomFct [(arg_1 [..., arg_n)]] RETURN type IS

[Déclarations locales]

BEGIN **Avec arg_i [IN / OUT / IN OUT] type [{ := | DEFAULT } expression]**

... corps de la fonction

RETURN ...

...

[EXCEPTION

... gestion des exceptions]

END nomFct ;

- ***Utilisation***

nomVar := ***nomFct*** *(…)*

- **PAQUETAGES**

CREATE PACKAGE nomPck **IS**

[déclaration de variables]

[déclaration procédures& fonctions]

END [nomPck] ;

CREATE PACKAGE BODY nomPck **IS**

[déclaration de variables]

[entête + corps de procédures et/ou fonctions]
…

END [nomPck] ;

- ***Utilisation***

*:**nomPck**.ressource (…)*

- **ET AUSSI**

CREATE [OR REPLACE]	Applicable aux fonctions, procédures… Crée en supprimant la version antérieure
SQLPLUS, activer l'affichage	SET SERVEROUTPUT ON
SQLPLUS, afficher dans une session	Package système DBMS_OUTPUT et ses fonctions comme : PUT_LINE (chaineCaractere)
SQL'PLUS, écrire un bloc PL/SQL	En le commençant par DECLARE, BEGIN ou CREATE selon la situation, et en le terminant par une ligne avec pour premier et unique caractère **/**
SHOW ERRORS	Voir les erreurs en cours, *à exécuter par exemple après l'écriture d'une procédure*
Le déclencheur (ou trigger)	Code déclenché quand l'évènement associé se produit. Le déclencheur est écrit en PL/SQL, *voir syntaxe additionnelle.*