

# Chapitre 8

## Manipulation avec Oracle

*"Le savant sait qu'il ignore."*  
- V. Hugo

### 1. Oracle et SQL

Norme SQL

Types de données reconnus par Oracle

### 2. Oracle et PL/SQL

Introduction

Architecture d'un programme PL/SQL

Section Declare

Structures de contrôle

Gestion des erreurs

Gestion des curseurs

### 3. Catalogue d'Oracle

# Oracle et SQL

## Norme SQL

- ◆ Oracle respecte la norme SQL 2
- ◆ Oracle offre des extensions à cette norme :
  - ◆ expressivité des requêtes de sélection
  - ◆ contrôle de l'intégrité des données
  - ◆ contrôle de la sécurité
  - ◆ gestion transactionnelles

## Types de données reconnus par Oracle

Type Oracle	Description
char( <i>taille</i> )	type caractère de longueur fixe, avec un maximum de 2000 caractères
varchar2( <i>taille</i> )	type caractère de longueur variable, avec un maximum de 4000 caractères
varchar	actuellement identique au type char
number( <i>précision, étendue</i> )	type numérique <ul style="list-style-type: none"><li>◆ <i>précision</i> : nombre total de chiffres</li><li>◆ <i>étendue</i> : nombre total de chiffres après la virgule</li></ul>
blob	grand objet binaire ( <i>binary large object</i> ) d'une taille maximale de 4 gigaoctets
date	date de longueur fixe (format Oracle : DD-MON-YY)
long	type caractère de longueur variable pouvant atteindre 2 gigaoctets

# Oracle et PL/SQL

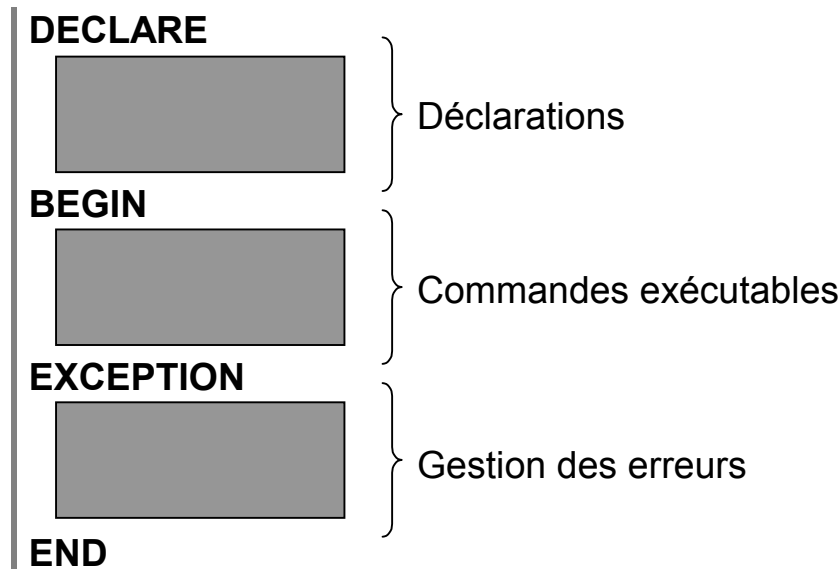
## Introduction

- ◆ PL/SQL est une extension procédurale du langage SQL
- ◆ PL/SQL permet de grouper un ensemble de requêtes SQL; de définir des contraintes d'intégrité complexes (*Triggers*); d'écrire des procédures stockées (*Stored Procedures*)
- ◆ PL/SQL =     SQL  
                  + opérateurs  
                  + variables  
                  + boucles  
                  + **gestion des curseurs**  
                  + **gestion des erreurs**
- ◆ Ici : juste les principes !
- ◆ Ailleurs : lire le livre de S. Feuerstein & al : "*Oracle PL/SQL – précis et concis* ", O'Reilly, Paris, 2000
- ◆ Ailleurs également : M. Abbey : "*Oracle 8i – notions fondamentales*", Oracle Press, Campus Press, Paris, 2000

# Oracle et PL/SQL

## Architecture d'un programme PL/SQL

- ◆ Un programme PL/SQL est composé de trois parties :



- ◆ Description

- ◆ La partie **Déclarations** sert à définir les variables et les constantes utilisées dans ce bloc. Cette partie commence par le mot clé **DECLARE**
- ◆ Les parties **Commandes** et **Exception** constituent le corps du bloc PL/SQL. Ce bloc doit commencer par le mot **BEGIN** et se terminer par **END**. Il est composé des éléments suivants :
  - Commandes SQL
  - Opérateurs de comparaison
  - Instructions de contrôle conditionnel
  - Instructions de contrôle itératif
  - Gestion des curseurs
  - Gestion des erreurs
- ◆ Une instruction est terminée par un ;

# Oracle et PL/SQL

## Section declare

- ◆ Section réservée à la déclaration des variables
- ◆ Plusieurs types de données prédéfinis. Les types les plus souvent utilisés :

Type Oracle	Description
<code>varchar2(<i>taille</i>)</code>	type alphanumérique de longueur variable (longueur maximale : 32 767 octets)
<code>number(<i>précision</i>, <i>étendue</i>)</code>	type numérique <ul style="list-style-type: none"><li>◆ <i>précision</i> : nombre total de chiffres</li><li>◆ <i>étendue</i> : nombre total de chiffres après la virgule</li></ul>
<code>date</code>	date de longueur fixe (format Oracle : DD-MON-YY)
<code>boolean</code>	type booléen (true, false, null)

- ◆ Exemple

```
DECLARE  
nombre          number;  
date_debut      date;  
nom             varchar2(30);  
moyenne         number(2,2);
```

# Oracle et PL/SQL

## Requêtes SQL

- ◆ Deux types de requêtes SQL dans un bloc PL/SQL :
  - ◆ requêtes simples (une seule ligne comme résultat de requête)
  - ◆ requêtes complexes (plus d'une ligne dans le résultat de requête ⇒ définition de curseurs – cfr. après)

### Requêtes simples

- ◆ Syntaxe *adaptée* de SFW

```
SELECT liste_de_sélection INTO liste_de_variables
FROM liste_de_tables
[condition_de_recherche]
```

- ◆ Description
  - ◆ la clause INTO permet de *lier* les colonnes du résultat avec les variables d'un bloc PL/SQL
  - ◆ *liste\_de\_variables* et *liste\_de\_sélection* doivent avoir le même nombre d'éléments et leurs types doivent correspondre un à un

- ◆ Exemple

```
DECLARE
    nombre          NUMBER(2);

BEGIN
    SELECT count(*) INTO nombre
    FROM eleves
    WHERE annee = 1
END
```

# Oracle et PL/SQL

## Structures de contrôle (1/3)

- ◆ Trois structures de contrôle sont définies :
  - ◆ **Structure sélective** teste une condition, puis exécute une séquence d'instructions à la place d'une autre selon que la condition évaluée est vraie (true) ou fausse (false)
  - ◆ **Structure itérative** exécute une suite d'instructions de façon répétée jusqu'à ce que la condition spécifiée soit rencontrée
  - ◆ **Structure séquentielle** exécute simplement une suite d'instructions dans l'ordre où elles ont été définies

### Structure sélective

- ◆ Test de conditions au moyen de trois structures logiques :
  - ◆ IF-THEN
  - ◆ IF-THEN-ELSE
  - ◆ IF-THEN-ELSIF

- ◆ Exemples

```
IF nombre > 10 THEN
    nombre := nombre + 20;
END IF;
```

```
IF nombre > 10 THEN
    nombre := nombre + 20;
ELSE
    nombre := nombre + 10;
END IF;
```

```
IF nombre > 10 THEN
    nombre := nombre + 20;
ELSIF nombre < 0 THEN
    nombre := nombre + 10;
END IF;
```

# Oracle et PL/SQL

## Structures de contrôle (2/3)

### Structure itérative (1/2)

- ◆ Répétition d'une série d'instructions. 4 types de structure itérative :
  - ◆ LOOP-EXIT-END
  - ◆ LOOP-EXIT-WHEN-END
  - ◆ WHILE-LOOP-END
  - ◆ FOR-IN-LOOP-END

- ◆ Exemples

boucle {

```
compteur :=1 ;  
  
LOOP  
    compteur:=compteur + 1;  
    IF compteur >= 100 THEN  
        EXIT;  
    END-IF;  
    ...  
END LOOP;  
...
```

sortie de boucle

boucle {

```
compteur :=1 ;  
  
LOOP  
    compteur:=compteur + 1;  
    EXIT WHEN compteur >= 100;  
    ...  
END LOOP;  
...
```

sortie de boucle



# Oracle et PL/SQL

## Structures de contrôle (3/3)

### Structure itérative (2/2)

#### ♦ Exemples (*suite*)

```
compteur :=1 ;  
  
boucle { WHILE compteur < 100 LOOP  
        ...  
        compteur:=compteur + 1;  
        ...  
        END LOOP;  
        ...  
  
boucle { FOR compteur IN 1 .. 100 LOOP  
        ...  
        END LOOP;  
        ...
```

# Oracle et PL/SQL

## Gestion des erreurs (exceptions) (1/3)

- ◆ Section optionnelle contenant le code nécessaire pour gérer les conditions d'erreurs
- ◆ Principe : chaque fois qu'une erreur se produit durant l'exécution d'un bloc PL/SQL, le contrôle passe automatiquement à la section EXCEPTION
- ◆ Deux types d'erreurs :
  - ◆ erreurs (ou exceptions) internes (définies par Oracle)
  - ◆ erreurs (ou exceptions) externes (définies par l'utilisateur)

### Erreurs internes

- ◆ Erreurs produites lorsqu'un bloc PL/SQL viole une règle d'Oracle ou dépasse une limite dépendant du système d'exploitation
- ◆ Erreurs courantes en PL/SQL :

Exception	Description
no_data_found	exception générée lorsqu'une instruction SELECT ne retourne aucune ligne
too_many_rows	exception générée lorsqu'une instruction SELECT retourne plus d'une ligne
dup_val_on_index	exception générée lorsqu'un bloc tente d'enregistrer une valeur dupliquée dans une colonne de table sur laquelle un index unique a été défini

# Oracle et PL/SQL

## Gestion des erreurs (exceptions) (2/3)

### Erreurs internes (suite)

Exception	Description
value_error	exception générée lorsqu'une opération d'affectation tente de placer une valeur dans un champ qui n'est pas assez long pour la contenir

### Erreurs externes

- ◆ Erreurs déclarées par l'utilisateur dans la partie déclarative d'un bloc PL/SQL
- ◆ Erreurs explicitement déclenchées par la commande RAISE
- ◆ La commande RAISE arrête l'exécution normale du bloc PL/SQL
- ◆ Exemple

```
DECLARE
    trop_eleves          EXCEPTION;
    nombre                NUMBER(2);

BEGIN
    SELECT count(*) INTO nombre
    FROM eleves
    WHERE annee = 1;

    IF nombre > 50 THEN
        RAISE trop_eleves
    END IF;

END
```

# Oracle et PL/SQL

## Gestion des erreurs (exceptions) (3/3)

### Traitement des erreurs internes et externes

- ◆ Lorsqu'une erreur est rencontrée, elle est interceptée par la structure conditionnelle WHEN-THEN-OTHERS :

```
WHEN type_exception_1 THEN gestion_erreur_1 ;  
{WHEN type_exception_2 THEN gestion_erreur_2 ;}  
[OTHERS gestion_autres_exceptions ;]
```

- ◆ Exemple

```
DECLARE  
    numero          NUMBER(2);  
    eleve_existe     BOOLEAN;  
  
BEGIN  
    eleve_existe := TRUE;  
  
    SELECT num INTO numero  
    FROM eleves  
    WHERE annee = 1  
    AND num_eleve IN (SELECT num_eleve  
                      FROM activites_pratiquees  
                      WHERE nom = 'Volley ball');  
  
    EXCEPTION  
    WHEN no_data_found THEN eleve_existe := FALSE;  
    OTHERS eleve_existe := NULL;  
  
END
```

si résultat vide



# Oracle et PL/SQL

## Gestion des curseurs (1/5)



### Notion de curseurs

- ◆ Espèce de pointeur permettant de parcourir un ensemble ordonné de lignes d'un résultat
- ◆ Lorsqu'un curseur C pointe vers la ligne L, on dit que C est **positionné** sur la ligne L
- ◆ Un curseur est toujours associé à une **expression de sélection** spécifiée dans l'instruction qui définit le curseur
- ◆ Exemple

Le curseur `celeve` est associé à l'expression de sélection des élèves de première année :

```
celeve ≈ SELECT nom  
        FROM eleves  
        WHERE annee=1;
```

Num	nom	prenom	date_naissance	Poids	annee
1	Brisefer	Benoit	10-12-1978	35	1
2	Génial	Olivier	10-04-1978	42	1
5	Tsuno	Yoko	29-10-1977	45	1
7	Lagaffe	Gaston	08-04-1975	61	1
9	Walthéry	Natacha	07-09-1977	59	1

# Oracle et PL/SQL

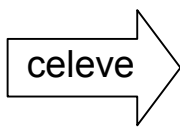
## Gestion des curseurs (2/5)

### Manipulation de curseurs

- ◆ Avant de pouvoir accéder aux lignes du résultat, le curseur associé doit être ouvert
- ◆ L'effet de cette commande est d'évaluer l'expression de sélection associée. Ensuite elle place le curseur dans l'état ouvert et le positionne avant la première ligne du résultat

- ◆ Exemple

| **OPEN** celeve ;

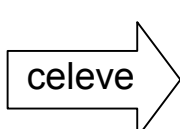


Num	nom	prenom	date_naissance	Poids	annee
1	Brisefer	Benoit	10-12-1978	35	1
2	Génial	Olivier	10-04-1978	42	1
5	Tsuno	Yoko	29-10-1977	45	1
7	Lagaffe	Gaston	08-04-1975	61	1
9	Walthéry	Natacha	07-09-1977	59	1

- ◆ Dès qu'un curseur est ouvert, il est possible de parcourir l'ensemble résultat au moyen de la commande FETCH

- ◆ Exemple

| **FETCH** celeve ;



Num	nom	prenom	date_naissance	Poids	annee
1	Brisefer	Benoit	10-12-1978	35	1
2	Génial	Olivier	10-04-1978	42	1
5	Tsuno	Yoko	29-10-1977	45	1
7	Lagaffe	Gaston	08-04-1975	61	1
9	Walthéry	Natacha	07-09-1977	59	1

# Oracle et PL/SQL

## Gestion des curseurs (3/5)

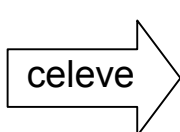
### Manipulation de curseurs

- ◆ Il existe une dernière opération sur les curseurs : il s'agit de la fermeture d'un curseur
- ◆ Lorsqu'un curseur est fermé, il est impossible d'accéder aux lignes qui lui étaient associés

### Application principale de curseurs

- ◆ Liaison entre les données de la base de données et les variables d'un programme d'application (C, COBOL, PL/SQL, etc)
- ◆ Illustration

Le curseur *celeve* pointe la première ligne du résultat de la requête :

 celeve	Num	nom	prenom	date_naissance	Poids	annee
	1	Brisefer	Benoit	10-12-1978	35	1
	2	Génial	Olivier	10-04-1978	42	1
	5	Tsuno	Yoko	29-10-1977	45	1
	7	Lagaffe	Gaston	08-04-1975	61	1
	9	Walthéry	Natacha	07-09-1977	59	1

Les valeurs de la ligne associées à ce curseur vont pouvoir être affectées à des variables d'un programme d'application

- ◆ Utilisation des curseurs dans PL/SQL (cfr. après), dans *Embedded SQL*, dans ODBC et dans JDBC (cfr. Chapitre 10)

# Oracle et PL/SQL

## Gestion des curseurs (4/5)

### Curseur et PL/SQL

- ◆ Un curseur est défini comme n'importe quelle variable PL/SQL
- ◆ Il est donc défini dans la section DECLARE d'un bloc PL/SQL et doit contenir **uniquement** des instructions SELECT
- ◆ Opérations sur les curseurs :
  - ◆ Définition et nomination des curseurs
  - ◆ Préparation (ou ouverture) du curseur pour l'utiliser
  - ◆ Récupération des données en utilisant un curseur
  - ◆ Libération de la mémoire occupée par un curseur lorsqu'il n'est plus utilisé
- ◆ Codage en 4 parties :
  - ◆ Définition du curseur dans la section DECLARE ①
  - ◆ Ouverture du curseur après le mot clé BEGIN initial du bloc PL/SQL ②
  - ◆ Récupération des données dans une ou plusieurs variables. L'instruction FETCH doit comporter le même nombre de variables réceptrices qu'il y a de colonnes dans la liste SELECT du curseur ③
  - ◆ Fermeture du curseur au moyen de l'instruction CLOSE ④
- ◆ Principaux paramètres d'un curseur :
  - ◆ %found ou %notfound : booléen indiquant si l'exécution du curseur a une ligne correspondant au critère de sélection
  - ◆ %rowcount: total des lignes du résultat



# Oracle et PL/SQL

## Gestion des curseurs (5/5)

### ◆ Exemple

```
DECLARE
    nom_1    VARCHAR2(30);

①  CURSOR c_eleve_1 IS
    SELECT nom
    FROM   eleves
    WHERE  annee=1;

BEGIN

②  OPEN c_eleve_1;
③  FETCH c_eleve_1 INTO nom_1;

    WHILE c_eleve_1%found LOOP
        ...
        FETCH c_eleve_1 INTO nom_1;
    END LOOP;

④  CLOSE c_eleve_1;
    ...
END
```

# Catalogue d'Oracle

## Présentation du catalogue d'Oracle

- ◆ Catalogue d'Oracle : base de données pouvant être accédée à travers le langage SQL
- ◆ Catalogue mis à jour automatiquement par le noyau d'Oracle
- ◆ Catalogue accessible uniquement en consultation
- ◆ Quatre types d'informations
  - ◆ informations relatives aux objets d'un utilisateur
  - ◆ informations relatives aux objets accessibles à un utilisateur
  - ◆ informations relatives aux administrateurs (*pas ici*)
  - ◆ informations relatives au suivi des performances (*pas ici*)

## Informations relatives aux objets d'un utilisateur

- ◆ Ensemble de tables donnant la description des objets appartenant à un utilisateur tels que les tables, les vues, etc.
- ◆ Le nom de ces tables commence par le préfixe "USER\_"
- ◆ Echantillon de ces tables :

Table	Contenu
USER_CATALOG	tables, vues, synonymes détenus par l'utilisateur
USER_CUSTERS	description des clusters de l'utilisateur
USER_CLU_COLUMNS	correspondances de colonnes de table et colonnes de cluster
USER_INDEXES	description des index de l'utilisateur

# Catalogue d'Oracle

## Informations relatives aux objets d'un utilisateur

Table	Contenu
USER_IND_COLUMNS	colonnes d'index utilisateur ou sur des tables utilisateur
USER_SYNONYMS	synonymes privés de l'utilisateur
USER_TABLE	description des tables de l'utilisateur
USER_TRIGGERS	description des déclencheurs de base de données créés par l'utilisateur, incluant le code source (cfr. Chapitre 9)
USER_TYPES	description des types de données définis par l'utilisateur et détenus par lui
USER_VIEWS	Description des vues de l'utilisateur (cfr. Chapitre 9)

## Informations relatives aux objets accessibles à un utilisateur

- ◆ Ensemble de tables donnant une description des objets appartenant à un utilisateur **et également** des objets auxquels un utilisateur est autorisé à accéder
- ◆ Le nom de ces tables commence par le préfixe "ALL\_"

# Catalogue d'Oracle

Informations relatives aux objets accessibles à un utilisateur

◆ Echantillon de ces tables :

Table	Contenu
ALL_CATALOG	tables, vues, synonymes accessibles à l'utilisateur
ALL_INDEX	description des index sur les tables accessibles à l'utilisateur
ALL_IND_COLUMNS	colonnes d'index sur des tables accessibles à l'utilisateur
ALL_SYNONYMS	synonymes accessibles à l'utilisateur
ALL_TABLES	description des tables accessibles à l'utilisateur