Ecrire des curseurs explicites

Objectifs

A la fin de ce chapitre, vous pourrez :

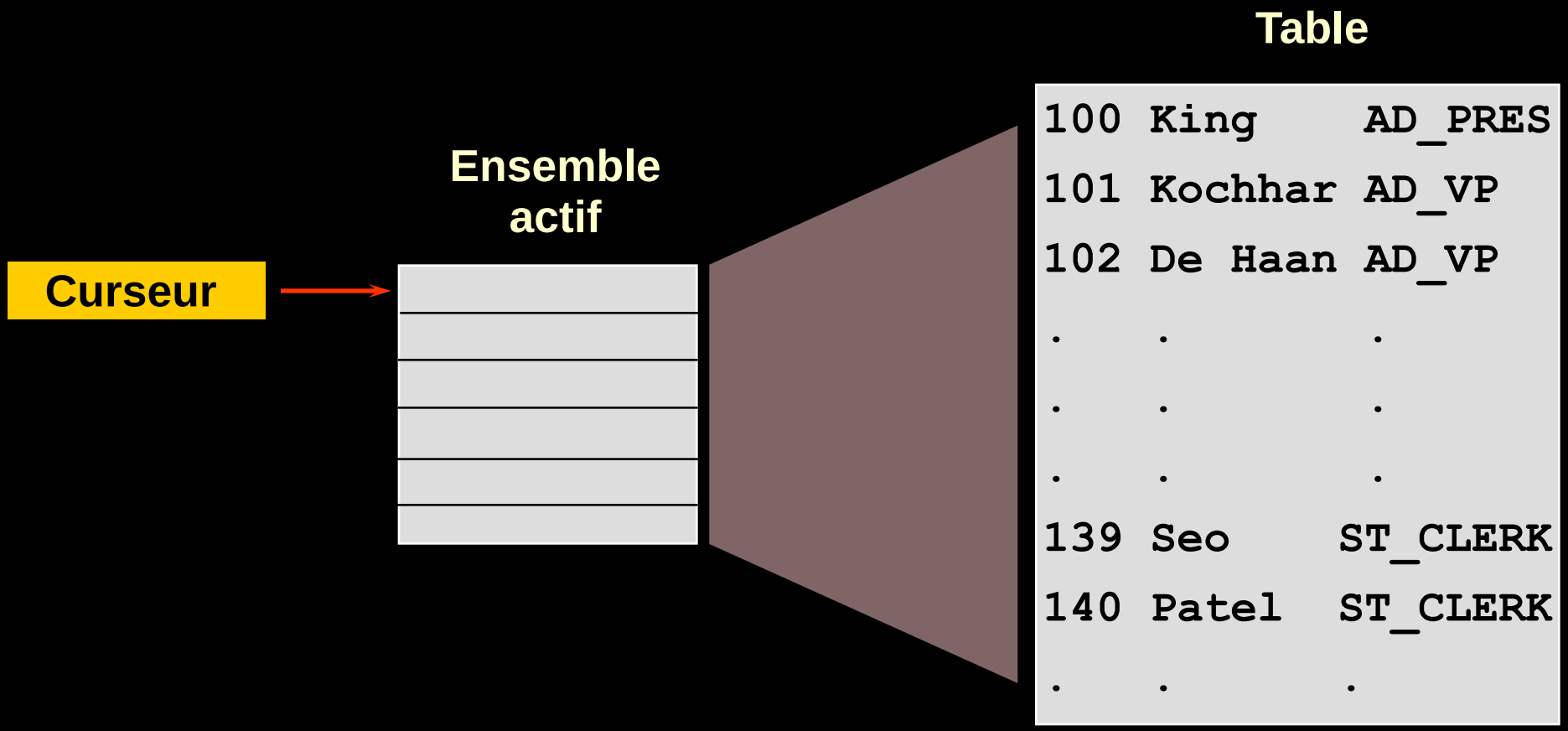
- faire la différence entre un curseur implicite et un curseur explicite
- savoir quand et pourquoi utiliser un curseur explicite
- utiliser une variable de type Record en PL/SQL
- écrire une boucle `FOR` de curseur

A propos des curseurs

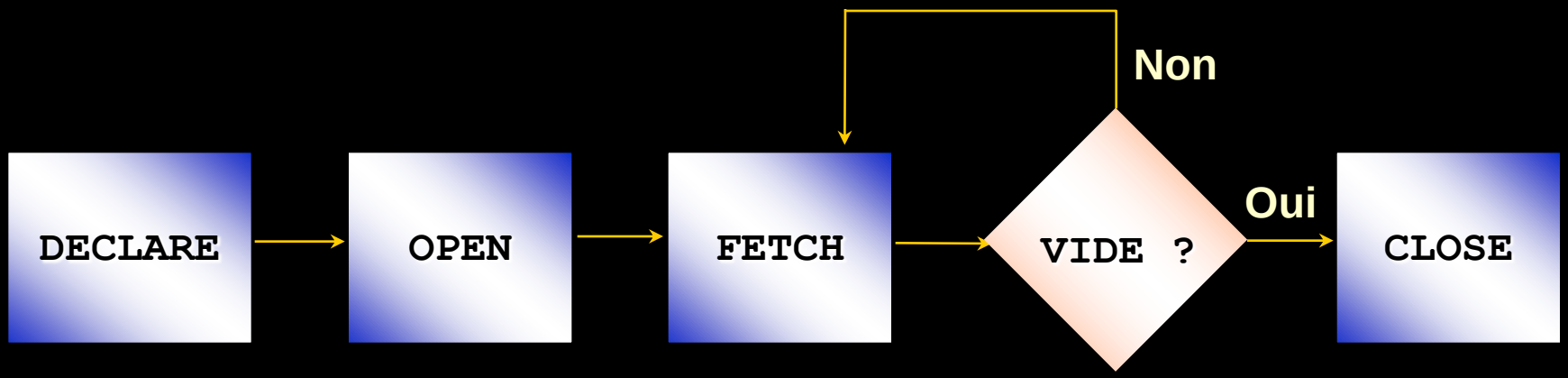
Chaque instruction SQL exécutée par le serveur Oracle a son propre curseur individuel qui lui est associé :

- curseurs implicites : déclarés pour toutes les instructions `SELECT` LMD et PL/SQL
- curseurs explicites : déclarés et nommés par le programmeur

Fonctions des curseurs explicites



Contrôler les curseurs explicites

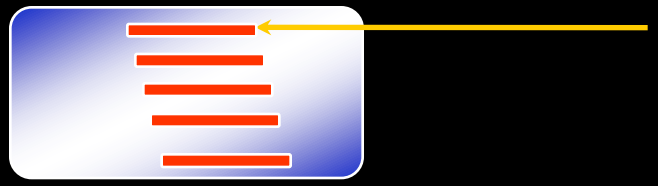


- Créer une zone SQL nommée
- Identifier l'ensemble actif
- Charger la ligne en cours dans des variables
- Tester l'existence de lignes
- Si des lignes existent, revenir à `FETCH`
- Libérer l'ensemble actif

Contrôler les curseurs explicites

1. Ouvrir le curseur
2. Extraire une ligne
3. Fermer le curseur

1. Ouvrir le curseur.

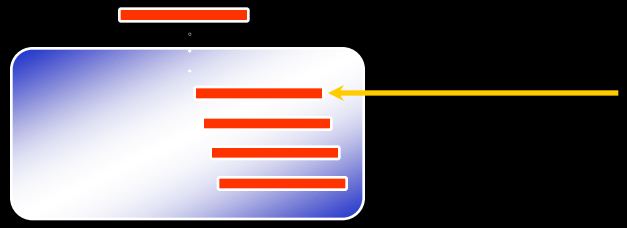


Pointeur
de curseur

Contrôler les curseurs explicites

1. Ouvrir le curseur
2. Extraire une ligne
3. Fermer le curseur

2. Extraire une ligne à l'aide du curseur.



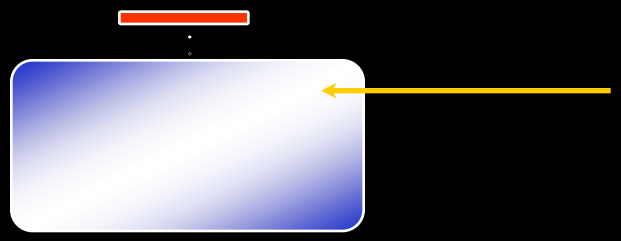
Pointeur
de curseur

Continuer jusqu'à ce que
le curseur soit vide

Contrôler les curseurs explicites

1. Ouvrir le curseur
2. Extraire une ligne
3. Fermer le curseur

3. Fermer le curseur.



Pointeur
de curseur

Déclarer le curseur

Syntaxe :

```
CURSOR cursor_name IS
      select_statement;
```

- **Ne pas inclure la clause `INTO` dans la déclaration du curseur**
- **S'il existe un ordre précis de traitement des lignes, utiliser la clause `ORDER BY` dans l'interrogation**

Déclarer le curseur

Exemple :

```
DECLARE
  CURSOR emp_cursor IS
    SELECT employee_id, last_name
    FROM   employees;

  CURSOR dept_cursor IS
    SELECT *
    FROM   departments
    WHERE  location_id = 170;
BEGIN
  ...
```

Ouvrir le curseur

Syntaxe :

```
OPEN  cursor_name;
```

- Ouvrir le curseur pour exécuter l'interrogation et identifier l'ensemble actif
- Si l'interrogation ne renvoie pas de ligne, aucune exception n'est déclenchée
- Utiliser les attributs de curseur pour tester le résultat après une extraction

Extraire les données à partir du curseur

Syntaxe :

```
FETCH cursor_name INTO  [variable1, variable2, ...]
                        | record_name];
```

- Extraire les valeurs de la ligne en cours pour les placer dans des variables
- Inclure le même nombre de variables
- Ajuster la position des variables par rapport aux colonnes
- Vérifier si le curseur contient des lignes.

Extraire les données à partir du curseur

Exemple :

```
LOOP
  FETCH emp_cursor INTO v_empno,v_ename;
  EXIT WHEN ...;
  ...
    -- Process the retrieved data
  …
END LOOP;
```

Fermer le curseur

Syntaxe :

```
CLOSE     cursor_name;
```

- **Fermer le curseur après avoir terminé le traitement des lignes**
- **Rouvrir le curseur, si nécessaire**
- **Ne pas essayer d'extraire les données d'un curseur s'il a été fermé.**

Attributs d'un curseur explicite

Obtenir les informations d'état concernant un curseur

Attribut	Type	Description
`%ISOPEN`	Booléen	Prend la valeur `TRUE` si le curseur est ouvert
`%NOTFOUND`	Booléen	Prend la valeur `TRUE` si la dernière extraction ne renvoie pas de ligne
`%FOUND`	Booléen	Prend la valeur `TRUE` si la dernière extraction renvoie une ligne ; complément de `%NOTFOUND`
`%ROWCOUNT`	Nombre	Prend la valeur correspondant au nombre total de lignes renvoyées jusqu'à présent

Attribut %ISOPEN

- Extraire les lignes uniquement lorsque le curseur est ouvert
- Utiliser l'attribut de curseur %ISOPEN avant de réaliser une extraction pour vérifier si le curseur est ouvert

Exemple :

```
IF NOT emp_cursor%ISOPEN THEN
    OPEN emp_cursor;
END IF;
LOOP
  FETCH emp_cursor...
```

Contrôler plusieurs extractions

- Traiter plusieurs lignes à partir d'un curseur explicite en utilisant une boucle
- Extraire une ligne à chaque itération
- Utiliser les attributs de curseur explicite pour contrôler le succès de chaque extraction.

Attributs %NOTFOUND et %ROWCOUNT

- Utiliser l'attribut de curseur %ROWCOUNT pour extraire un nombre exact de lignes
- Utiliser l'attribut de curseur %NOTFOUND pour déterminer dans quels cas la sortie de la boucle doit s'effectuer.

Exemple

```
DECLARE
      v_empno  employees.employee_id%TYPE;
      v_ename  employees.last_name%TYPE;
      CURSOR emp_cursor IS
        SELECT employee_id, last_name
        FROM   employees;
    BEGIN
      OPEN emp_cursor;
      LOOP
        FETCH emp_cursor INTO v_empno, v_ename;
        EXIT WHEN emp_cursor%ROWCOUNT > 10 OR
                          emp_cursor%NOTFOUND;
        DBMS_OUTPUT.PUT_LINE (TO_CHAR(v_empno)
                              ||'   '|| v_ename);
      END LOOP;
      CLOSE emp_cursor;
END ;
```

Curseurs et enregistrements

Traiter les lignes de l'ensemble actif en extrayant les valeurs pour les placer dans un enregistrement PL/SQL.

```
DECLARE
  CURSOR emp_cursor IS
    SELECT  employee_id, last_name
    FROM    employees;
  emp_record   emp_cursor%ROWTYPE;
BEGIN
  OPEN emp_cursor;
  LOOP
    FETCH emp_cursor INTO emp_record;
  ...
```

emp_record

employee_id	last_name
100	King

Boucles FOR de curseur

Syntaxe :

```
FOR record_name IN cursor_name LOOP
  statement1;
  statement2;
  . . .
END LOOP;
```

- La boucle FOR de curseur simplifie le traitement des curseurs explicites
- Des opérations d'ouverture, d'extraction, de sortie et de fermeture ont lieu de manière implicite
- L'enregistrement est déclaré implicitement.

Boucles FOR de curseur

Afficher une liste des employés travaillant dans le service des ventes.

```
DECLARE
  CURSOR emp_cursor IS
    SELECT last_name, department_id
    FROM   employees;
BEGIN
  FOR emp_record IN emp_cursor LOOP
         -- implicit open and implicit fetch occur
    IF emp_record.department_id = 80 THEN
      ...
  END LOOP; -- implicit close occurs
END;
/
```

Boucles FOR de curseur utilisant des sous-interrogations

Il n'est pas nécessaire de déclarer le curseur.

Exemple :

```
BEGIN
  FOR emp_record IN (SELECT last_name, department_id
                     FROM   employees) LOOP
        -- implicit open and implicit fetch occur
    IF emp_record.department_id = 80 THEN
      ...
  END LOOP; -- implicit close occurs
END;
```

Synthèse

Ce chapitre vous a permis d'apprendre à :

- différencier les types de curseur :
  - curseurs implicites : utilisés pour toutes les instructions `LMD` et les interrogations monolignes
  - curseurs explicites : utilisés pour les interrogations portant sur zéro, une ou plusieurs lignes
- manipuler les curseurs explicites
- évaluer l'état du curseur en utilisant des attributs de curseur
- utiliser les boucles `FOR` de curseur

Présentation de l'exercice 6

Dans cet exercice, vous allez :

- déclarer et utiliser des curseurs explicites pour interroger les lignes d'une table
- utiliser une boucle `FOR` de curseur
- appliquer des attributs de curseur pour contrôler l'état du curseur