# PL / SQL
# Les curseurs

**Ines BAKLOUTI**

ines.baklouti@esprit.tn

Ecole Supérieure Privée d'Ingénierie et de Technologies

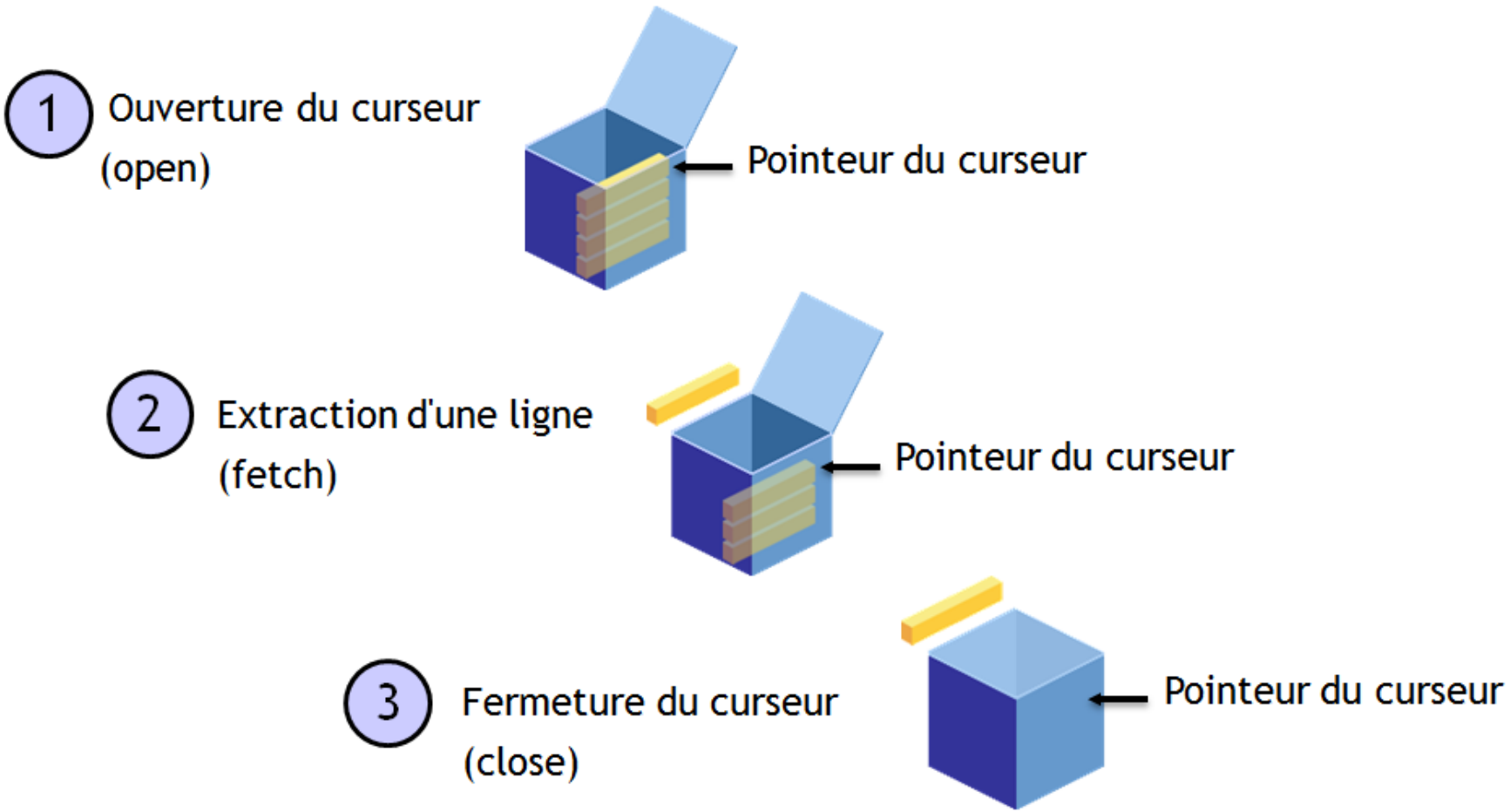
# Plan

# Introduction

- Un curseur est une variable qui pointe vers le résultat d'une requête SQL,
- Types de curseurs:
  - Curseurs implicites:
    - Déclarés implicitement et manipulés par SQL
    - Pour toute requête SQL du LMD et pour les interrogations qui retournent un seul enregistrement
  - Curseurs explicites:
    - Déclarés et manipulés par l'utilisateur
    - Pour les interrogations qui retournent plus qu'un enregistrement

# Plan

# Manipulation des curseurs



# Déclaration

## Syntaxe

**CURSOR** nom_curseur **IS** instruction select;

## Exemple

```
DECLARE
    CURSOR cur_emp IS
    SELECT employee_id, first_name, last_name FROM employees
    WHERE department_id = 100;

    CURSOR cur_dept IS
    SELECT * FROM departments
    ORDER BY department_id;
```

# Ouverture

## Syntaxe

**OPEN** nom_curseur;

- Ouvre le curseur pour exécuter la requête et identifier l'ensemble des lignes.
- Allocation mémoire pour y stocker le résultat.
- Si la requête ne retourne pas de ligne, aucune exception n'est signalée.

## Exemple

```
DECLARE
    CURSOR cur_emp IS
    SELECT employee_id, first_name, last_name FROM employees
    WHERE department_id = 100;
BEGIN
    OPEN cur_emp;
END;
```

# Exécution

## Syntaxe

**FETCH** nom_curseur **INTO** [variable1, variable2, . . . | record_name];

- Lecture de la zone pointée par le curseur.
- Affectation des valeurs de la ligne courante dans les variables de sortie (variables hôtes).
- Passage à l'enregistrement suivant.
- Il faut:
  - Prévoir le même nombre de variables
  - Testez si le curseur contient des lignes

# Exécution

## Exemple - version 1

```
DECLARE
    CURSOR cur_emp IS
    SELECT employee_id, first_name, last_name
    FROM employees
    WHERE department_id = 100;
    v_no employees.employee_id%TYPE;
    v_Fname employees.first_name%TYPE;
    v_Lname employees.last_name%TYPE;
BEGIN
    IF NOT cur_emp%ISOPEN THEN
        OPEN cur_emp;
    END IF;
    FETCH cur_emp INTO v_no, v_Fname,
    v_Lname;
    WHILE cur_emp%FOUND
    LOOP
    DBMS_OUTPUT.PUT_LINE('Employé no:
'||v_no||' Nom: '||v_Fname||'Prénom: '||v_Lname);
    FETCH cur_emp INTO v_no, v_Fname,
    v_Lname;
    END LOOP;
END;
```

## Exemple - version 2

```
DECLARE
    CURSOR cur_emp IS
    SELECT employee_id, first_name, last_name
    FROM employees
    WHERE department_id = 100;
    v_no employees.employee_id%TYPE;
    v_Fname employees.first_name%TYPE;
    v_Lname employees.last_name%TYPE;
BEGIN
    IF NOT cur_emp%ISOPEN THEN
        OPEN cur_emp;
    END IF;
    LOOP
    FETCH cur_emp INTO v_no, v_Fname,
    v_Lname;
    EXIT WHEN cur_emp%NOTFOUND;
    DBMS_OUTPUT.PUT_LINE('Employé no:
'||v_no||' Nom: '||v_Fname||'Prénom: '||v_Lname);
    END LOOP;
END;
```

# Fermuture

## Syntaxe

**CLOSE** nom_curseur;

- Fermeture du curseur.
- Libération de la zone mémoire allouée pour le résultat de la requête.

## Exemple

```
DECLARE
   CURSOR cur_emp IS
   SELECT employee_id FROM employees WHERE department_id = 100;
   v_no employees.employee_id%TYPE;
BEGIN
      OPEN cur_emp;
   LOOP
   FETCH cur_emp INTO v_no;
   EXIT WHEN cur_emp%NOTFOUND;
   DBMS_OUTPUT.PUT_LINE('Employé no: '||v_no);
   END LOOP;
   close cur_emp;
END;
```

# Plan

# Attributs des curseurs

- Chaque curseur possède quatre attributs:
  - **%ISOPEN** Génère un booléen VRAI lorsque le curseur spécifié en argument est ouvert.
  - **%FOUND** Génère un booléen VRAI lorsque le FETCH réussi (données lues).
  - **%NOTFOUND** Inverse de %FOUND (généralement plus utilisé que %FOUND).
  - **%ROWCOUNT** Renvoie le nombre de lignes contenues.
- Chaque attribut s'utilise en étant préfixé par le nom du curseur: nom_curseur%ATTRIBUT

# Attributs des curseurs

## Exemple

```
DECLARE
    CURSOR cur_emp IS SELECT employee_id, last_name FROM employees;
    v_no employees.employee_id%TYPE;
    v_Lname employees.last_name%TYPE;
BEGIN
    IF NOT cur_emp%ISOPEN THEN
    OPEN cur_emp;
    END IF;
    FETCH cur_emp INTO v_no, v_Lname;
    WHILE cur_emp%FOUND
    LOOP
    dbms_output.put_line(cur_emp%rowcount);
    FETCH cur_emp INTO v_no, v_Lname;
    END LOOP;
    CLOSE cur_emp;
END;
```

# Plan

# Utilisation simplifiée des curseurs

## Utilisation du type RECORD

- Déclarer implicitement un enregistrement (type record) dont les éléments sont de même type que les colonnes retournées par le curseur.

**Exemple**

```
DECLARE
CURSOR cur_emp IS
SELECT employee_id, last_name FROM employees;
rec_emp cur_emp%ROWTYPE;
```

- Pour utiliser les colonnes de l'enregistrement:
  - rec_emp.employee_id
  - rec_emp.last_name
- L'enregistrement est renseigné par le fetch:
  - Fetch cur_emp into rec_emp;

# Utilisation simplifiée des curseurs

## Utilisation du type RECORD

### Exemple

```
DECLARE
    CURSOR cur_emp IS SELECT employee_id, last_name FROM employees;
    rec_emp cur_emp%ROWTYPE;
BEGIN
    OPEN cur_emp;
    LOOP
    FETCH cur_emp INTO rec_emp;
    EXIT WHEN cur_emp%NOTFOUND;
    dbms_output.put_line('Employé no: '||rec_emp.employee_id||' Nom:
'||rec_emp.last_name);
    END LOOP;
    CLOSE cur_emp;
END;
```

# Utilisation simplifiée des curseurs

## Utilisation de la structure FOR .. IN

- La syntaxe **FOR .. IN** permet d'éviter de déclarer l'enregistrement hôte dans la partie DECLARE.

### Syntaxe

**FOR** nom_record **IN** nom_curseur
**LOOP**
  – ouverture automatique du curseur
  – fetch automatique
  – condition de sortie automatique
/* traitement */
**END LOOP;**
  – fermuture automatique du curseur
END;

- pas de déclaration de record
- On utilise les variables par nom_record.nom_variable par exemple: rec_emp.employee_id

# Utilisation simplifiée des curseurs

## Utilisation de la structure FOR .. IN

### Exemple

```
DECLARE
    CURSOR cur_emp IS SELECT employee_id, last_name FROM employees;
BEGIN
    FOR rec_emp IN cur_emp
    LOOP
    dbms_output.put_line('Employé no: '||rec_emp.employee_id||' Nom:
'||rec_emp.last_name);
    END LOOP;
END;
```

# Utilisation simplifiée des curseurs

## Utilisation de sous requête dans la structure FOR .. IN

- La syntaxe **FOR .. IN** avec utilisation de sous requête permet d'éviter de déclarer le curseur dans la partie DECLARE.

### Syntaxe

**FOR** nom_record **IN** requête select – déclaration implicite du record et du curseur
**LOOP**
– ouverture automatique du curseur
– fetch automatique
– condition de sortie automatique
/* traitement */
**END LOOP;** – fermuture automatique du curseur
END;

- pas de déclaration du curseur
- pas de déclaration de record
- On utilise les variables par nom_record.nom_variable par exemple: rec_emp.employee_id

# Utilisation simplifiée des curseurs

## Utilisation de sous requête dans la structure FOR .. IN

### Exemple

```
BEGIN
    FOR rec_emp IN (select employee_id, last_name from employees)
    LOOP
    dbms_output.put_line('Employé no: '||rec_emp.employee_id||' Nom:
'||rec_emp.last_name);
    END LOOP;
END;
```

# Plan

# Curseurs paramétrés

- Objectif : paramétrer la requête associée à un curseur pour éviter de multiplier les curseurs similaires.
- Syntaxe de définition: **Cursor** nom_curseur (param1 type, param2 type, ...) **IS** requête select;
- Les valeurs des paramètres sont transmises à l'ouverture du curseur: OPEN nom_curseur(valeurParam1, valeurParam2, ...);
- Il faut évidemment fermer le curseur avant de l'appeler avec d'autres valeurs pour les paramètres (sauf si on utilise la boucle for qui ferme automatiquement le curseur).

# Curseurs paramétrés

## Exemple

```
DECLARE
    CURSOR cur_emp(v_dept number,v_sal employees.salary%TYPE) IS SELECT
employee_id, last_name,salary FROM employees where department_id=v_dept and
salary>v_sal;
    rec_emp cur_emp%ROWTYPE;
BEGIN
    OPEN cur_emp(80,6000);
    LOOP
    FETCH cur_emp INTO rec_emp;
    EXIT WHEN cur_emp%NOTFOUND;
    dbms_output.put_line('Employé no: '||rec_emp.employee_id||' Nom:
'||rec_emp.last_name||' Salaire: '||rec_emp.salary);
    END LOOP;
    CLOSE cur_emp;
END;
```