

SGBD - 2^e

PL-SQL - Chapitre 9 - Les Curseurs

Daniel Schreurs

10 août 2022

Haute École de la Province de Liège

Table des matières du chapitre i

1. Introduction
2. Curseurs explicites
3. Curseurs for implicites
4. Curseurs for explicites
5. Curseurs avec paramètres
6. Curseurs for update
7. Le curseur implicite SQL

Introduction

- Les curseurs permettent de traiter les requêtes de recherche dont le résultat contient un nombre quelconque de tuples.
- La notion de curseur réalise le lien entre la philosophie ensembliste de SQL (set-retrieval) et celle, enregistrement par enregistrement (record-at-a-time) des langages de 3ème génération.

- Un curseur peut être considéré comme une espèce de pointeur¹ ;
- Un curseur est toujours associé à une expression de sélection.

1. Il peut être utilisé pour parcourir un ensemble ordonné de tuples (active set

Introduction

Exemple simple

```
1 DECLARE
2     CURSOR C -- Ici
3         IS SELECT nom, prenom
4           FROM clients
5          WHERE codepostal = '4000';
6 BEGIN
7     null;
8 END;
```

- L'expression de sélection n'est pas évaluée au moment de la déclaration
- L'expression de sélection est évaluée lorsque le curseur est ouvert
- À ce moment, l'ensemble des tuples du résultat de l'expression de sélection est associé au curseur.
- L'association se maintient jusqu'à la fermeture du curseur
- Le résultat de l'expression de sélection est figé jusqu'à la fermeture du curseur

Curseurs explicites

Curseurs explicites :

- Un curseur peut être déclaré dans un bloc PL/SQL, une procédure ou un package
`CURSOR nom_curseur IS expression_de_sélection;`
- Aucune restriction au niveau de la requête de sélection
- Un curseur DOIT ÊTRE déclaré avant d'être utilisé. Sinon :
exception `invalid_cursor`
- 3 opérations pour manipuler les curseurs : `open`, `fetch` et `close`

Curseurs explicites : Open

- `OPEN nom curseur;`
- L'effet du `OPEN` est d'évaluer l'expression de sélection associée.
- Le curseur est dans l'état ouvert et est positionné avant la première ligne du résultat
- Un curseur ouvert peut être parcouru par une série de `fetch`
- La tentative d'ouverture d'un curseur déjà ouvert provoque l'exception `cursor_already_open`

Curseurs explicites : FETCH

Récupérer les données

```
1 DECLARE
2     CURSOR C -- Ici
3         IS SELECT nom, prenom
4           FROM Clients
5           WHERE codepostal = '4000';
6 BEGIN
7     -- récupérer les données du curseur
8     FETCH nom_curseur
9         INTO [Liste_variable | record];
10 --OU
11 -- récupérer les données du curseur
12 FETCH nom_curseur
13     BULK COLLECT INTO collection;
14 END;
```

- Les variables de la liste ou les champs du record doivent être compatibles avec la clause select
- Idem si on utilise **BULK COLLECT INTO**

Curseurs explicites : CLOSE

- `CLOSE nom_curseur;`
- La fermeture d'un curseur non ouvert déclenche l'exception `invalid_cursor`

Curseurs explicites i

Exemple complet

```
1 DECLARE
2     CURSOR CurEmp IS -- 1
3         SELECT *
4         FROM Employes
5         WHERE nom LIKE 'C%';
6     UnEmploye Employes%ROWTYPE;
7 BEGIN
8     OPEN CurEmp; -- 2
9     FETCH CurEmp INTO UnEmploye; -- 3
10    WHILE CurEmp%FOUND
11        LOOP
12            DBMS_OUTPUT.PUT_LINE(UnEmploye.nom);
13            FETCH CurEmp INTO UnEmploye; -- 3
14        END LOOP; -- 4
```

Curseurs explicites ii

```
15      CLOSE CurEmp;  
16  EXCEPTION  
17      WHEN INVALID_CURSOR  
18          THEN DBMS_OUTPUT.PUT_LINE('Erreur Curseur CurEmp  
          ');  
19  END ;
```

Curseurs explicites i

Exemple complet 2

```
1 DECLARE
2     TYPE t_employe IS TABLE OF Employes %ROWTYPE;
3     CURSOR CurEmp IS -- 1
4         SELECT *
5         FROM Employes
6         WHERE nom LIKE 'C%';
7     TousLesEmployes t_employe;
8 BEGIN
9     OPEN CurEmp; -- 2
10    FETCH CurEmp BULK COLLECT INTO TousLesEmployes; -- 3
11    FOR i in 1..TousLesEmployes.COUNT
12        LOOP
13        DBMS_OUTPUT.PUT_LINE(TousLesEmployes(i).nom)
14            ; -- 3
```


Curseurs explicites ii

```
14         END LOOP;  
15     CLOSE CurEmp;  
16 EXCEPTION  
17     WHEN INVALID_CURSOR  
18         THEN DBMS_OUTPUT.PUT_LINE('Erreur Curseur CurEmp'  
19                                     ');  
19 END;
```

Chaque curseur possède 4 attributs :

- %FOUND vaut **TRUE** si un **FETCH** a pu extraire un tuple
- %NOTFOUND vaut **TRUE** si un **FETCH** n'a pas pu extraire de tuple
- %ISOPEN vaut **TRUE** si le curseur est ouvert
- %ROWCOUNT vaut le nombre de tuples qui ont déjà été extraits par des **FETCH**

Curseurs explicites

		%FOUND	%NOTFOUND	%ISOPEN	%ROWCOUNT
OPEN	Avant	Exception	Exception	FALSE	Exception
	Après	NULL	NULL	TRUE	0
1er FETCH	Avant	NULL	NULL	TRUE	0
	Après	TRUE	FALSE	TRUE	1
Nème FETCH	Avant	TRUE	FALSE	TRUE	*
	Après	TRUE	FALSE	TRUE	*
Dernier FETCH	Avant	TRUE	FALSE	TRUE	*
	Après	FALSE	TRUE	TRUE	*
CLOSE	Avant	FALSE	TRUE	TRUE	*
	Après	Exception	Exception	FALSE	Exception

Curseurs explicites

Légende du tableau :

- * signifie que le résultat dépend du nombre de **FETCH** exécutés
- Après le 1er **FETCH**, si le résultat est vide, **%FOUND** donne **FALSE**, **%NOTFOUND** donne **TRUE** et **%ROWCOUNT** donne 0
- **%ISOPEN** est souvent utilisé dans le traitement des exceptions pour fermer le ou les curseurs ouverts

```
1 BEGIN
2     null;
3 EXCEPTION
4     WHEN uneException THEN
5         IF c%ISOPEN THEN
6             CLOSE c; -- On ferme le curseur
7         END IF;
8 END;
```

Curseurs for implicites

Curseurs for implicites

Du point de vue programmation : ce sont les plus simples ! Il suffit de placer la requête de sélection.

```
1 BEGIN
2     FOR Employee IN
3         (SELECT *
4           FROM Emp
5           WHERE job LIKE 'A%')
6     LOOP
7         DBMS_OUTPUT.PUT_LINE(Employee.nom);
8     END LOOP;
9 END;
```

Curseurs for explicites

- Le curseur est déclaré de manière explicite;
- Son parcours est réalisé au moyen d'une boucle `for` et d'un record ad hoc implicitement déclaré dans la boucle;
- Il est possible d'utiliser les attributs `%FOUND`, `%NOTFOUND`, `%ISOPEN`, `%ROWCOUNT`.

Curseurs for explicites

Curseurs for explicites

```
1 DECLARE
2     CURSOR lesEmployes IS
3         SELECT *
4         FROM Emp
5         WHERE sal > 2500;
6 BEGIN
7     FOR unEmploye IN lesEmployes
8     LOOP
9         DBMS_OUTPUT.PUT_LINE(lesEmployes%ROWCOUNT
10                                || ' ' || unEmploye.ename
11                                || ' ' || unEmploye.job);
12     END LOOP;
13 END;
```

Curseurs avec paramètres

- Il est possible de passer des paramètres à un curseur
- La déclaration des paramètres dans le curseur se fait lors de la déclaration du curseur, de la même manière que pour une procédure
- L'évaluation et la transmission des paramètres se fait au moment de l'ouverture du curseur

Curseurs avec paramètres

Curseurs avec paramètres

```
1 CREATE OR REPLACE PROCEDURE EmployeeJob
2     (Job Emp.Job%TYPE) AS
3     CURSOR lesEmployes (x IN Emp.Job%TYPE) IS -- param x
4         SELECT *
5         FROM Emp
6         WHERE job LIKE x || '%'; -- param x
7 BEGIN
8     FOR unEmploye IN lesEmployes (EmployeeJob.Job)
9     LOOP
10         DBMS_OUTPUT.PUT_LINE (unEmploye.ename);
11     END LOOP;
12 EXCEPTION
13     WHEN OTHERS THEN RAISE;
14 END;
15
16 EXECUTE EmployeeJob('A');
```

Curseurs avec paramètres i

Curseurs avec paramètres

```
1 CREATE OR REPLACE PROCEDURE EmployeJob(Job Emp.Job%TYPE)
  AS
2     CURSOR lesEmployes (x IN Emp.Job%TYPE) IS -- param x
3         SELECT *
4         FROM Emp
5         WHERE job LIKE x || '%';
6     UnEmploye Emp%ROWTYPE;
7 BEGIN
8     OPEN lesEmployes(EmployeJob.Job);
9     FETCH lesEmployes INTO unEmploye
10    WHILE lesEmployes%FOUND
11        LOOP
12        DBMS_OUTPUT.PUT_LINE(lesEmployes%ROWCOUNT ||
```

Curseurs avec paramètres ii

```
13         ' ' || unEmploye.ename)
           ;
14     FETCH lesEmployes INTO UnEmploye;
15     END LOOP;
16     CLOSE lesEmployes;
17 END;
```

Curseurs for update

Curseurs for update

- Les curseurs for update permettent de modifier le tuple qui vient d'être extrait par un **FETCH**
- Oracle place des verrous exclusifs sur les tuples de l'active set des curseurs for update

```
1 SELECT ... FROM ...  
2 FOR UPDATE [OF liste_colonnes]  
3 [WAIT/NOWAIT]
```


Le curseur implicite SQL

- PL/SQL utilise un curseur implicite pour chaque opération du LMD de SQL (**INSERT**, **UPDATE**, **DELETE**);
- Ce curseur porte le nom SQL et il est exploitable après avoir exécuté l'instruction;

Le curseur implicite SQL :

- Ne nécessite pas de déclaration ;
- Ne doit pas être ouvert, ni fermé ;
- Peut utiliser les attributs %FOUND, %NOTFOUND, %ISOPEN, %ROWCOUNT ;
- Permet d'exécuter des INSERT, UPDATE, DELETE, SELECT INTO d'un seul tuple²

2. Contrairement aux autres curseurs : recherche portant sur plusieurs tuples.

Le curseur implicite SQL

Mise à jour ou insertion

```
1 BEGIN
2     UPDATE Emp
3     SET Job = 'A. S.'
4     WHERE ename = 'BOND';
5     IF SQL%NOTFOUND THEN -- NOTFOUND
6         INSERT INTO Emp
7             (EmpNo, Ename, job, HireDate, sal)
8             VALUES ('007', 'BOND', 'A. S.', CURRENT_DATE,
9                     10000);
9     END IF;
10 END;
```

Le curseur implicite SQL i

Effacer le projet si ce projet a commencé il y plus de trois ans

```
1 DECLARE
2     ExcIntRef EXCEPTION;
3     PRAGMA EXCEPTION_INIT (ExcIntRef,-2292);
4 BEGIN
5     DELETE
6     FROM emppro p1
7     WHERE p1.numpro = 'p10352'
8         AND 36 < (SELECT MONTHS_BETWEEN(CURRENT_DATE, p2.
9                     DATEDebut)
10                     FROM projets p2
11                     WHERE p1.numpro = p2.numpro);
12 DELETE FROM projets WHERE numpro = 'p10352';
13 IF SQL%NOTFOUND THEN -- Ici
```

Le curseur implicite SQL ii

```
13         DBMS_OUTPUT.PUT_LINE('Le nr de projet p10352 n''
           existe pas ');
14     ELSE
15         DBMS_OUTPUT.PUT_LINE('Le nr de projet p10352 est
           effacé ');
16     END IF;
17 EXCEPTION
18     WHEN ExcIntRef THEN DBMS_OUTPUT.PUT_LINE('Le projet
           ne peut être effacé');
19 END;
```