

Chapitre 9 : Les curseurs Oracle ©

INF3080 BASES DE DONNÉES (SGBD)

Guy Francoeur

Aucune reproduction sans autorisation

3 septembre 2019

UQÀM | **Département d'informatique**

- ▶ Les droits de lecture sont accordés aux étudiants inscrits au cours INF3080-030 A2019 uniquement;
- ▶ Aucun droit pédagogique ou reproduction n'est accordé sans autorisation;

Table des matières

1. curseur

2. boucle

Table des matières

1. curseur

- définition

- curseur simple

- curseur avec paramètre

2. boucle

Un curseur (*Cursor*) en Oracle est une variable qui est construite à partir d'un SELECT. Elle (variable) contient entre 0 et n tuples et m colonnes.

```
CURSOR cursor_name  
IS  
    SELECT 1 FROM dual  
    UNION SELECT 7 FROM dual;
```

```
CURSOR c1  
IS  
    SELECT pLangue, cLangue  
    FROM Langue  
    WHERE pLangue < 10;
```

curseur - déclaration

```
CREATE OR REPLACE FUNCTION f_A (icVal IN varchar2)
  RETURN number
IS
  vnLangue number;

  CURSOR C1 IS
    SELECT pLangue
    FROM Langue
    WHERE pLangue < 10;
BEGIN

  ... code pl/sql ...

  RETURN vnLangue;
END;
/
```

curseur - avec paramètre

Il est possible de faire des curseurs avec en fournissant des valeurs a celui afin de le rendre légèrement plus dynamique.

```
CREATE PROCEDURE p_A
AS
    CURSOR c2 (inLangue IN number) IS
        SELECT pClient, cClient
        FROM Client
        WHERE pLangue = inLangue;
BEGIN

    ... pl/sql code ...

EXCEPTION
    WHEN OTHERS THEN
        DBMS_OUTPUT.PUT_LINE(sqlerrm);
END p_A;
/
```

► OPEN, FETCH, NOTFOUND, FOUND, CLOSE

```
CREATE OR REPLACE FUNCTION f_A (icVal IN varchar2)
  RETURN number
IS
  vnLange number;

  CURSOR C1 IS
    SELECT pLange FROM Langu WHERE pLange < 10;
BEGIN
  OPEN C1;
  FETCH C1 INTO vnLange;

  if C1%notfound then
    vnLange := 0;
  end if;

  CLOSE C1;

  RETURN vnLange;
END;
/
```


1. curseur

2. boucle

- définition

- boucle simple

- boucle sur le curseur

- gestion d'erreur

Boucle - définition

Le langage pl/sql est similaire à plusieurs autres langages que vous avez déjà utilisés. Il offre aussi les boucles.

- ▶ `LOOP ... END LOOP;`
- ▶ `FOR variable IN [REVERSE] n..m LOOP ... END LOOP;`
- ▶ `WHILE condition LOOP ... END LOOP;`
- ▶ curseur `FOR LOOP`

Boucle - exemple

```
LOOP
  vnVal := vnVal + 2;
  EXIT WHEN vnVal > 20;
END LOOP;
```

```
FOR i IN 1..20
LOOP
  vnVal := i + 3;
END LOOP;
```

```
WHILE vnVal < 200
LOOP
  vnVal := i + 3;
END LOOP;
```

curseur et boucle

```
CREATE OR REPLACE FUNCTION f_B(inCommande IN number)
RETURN number
IS
    vnTotal number(6,2);

    cursor C2 IS
        SELECT nPrix
        FROM CommandeD
        WHERE CommandeE=inCommandeE;
BEGIN
    vnTotal := 0;

    FOR CommandeD_rec in C2
    LOOP
        vnTotal := vnTotal + CommandeD_rec.nPrix;
    END LOOP;

    RETURN vnTotal;
END;
```

curseur - FETCH

```
DECLARE
    v_trip_id      business_trips.bt_id_pk%TYPE;
    v_hotel_id     business_trips.bt_hotel_id%TYPE;
    CURSOR trip_cursor IS
        SELECT bt_id_pk, bt_hotel_id
        FROM business_trips;
BEGIN
    OPEN trip_cursor;
    LOOP
        FETCH trip_cursor INTO v_trip_id, v_hotel_id;
        EXIT WHEN
            trip_cursor%ROWCOUNT > 20
            OR trip_cursor%NOTFOUND;
        ... code pl/sql ...
    END LOOP;
    CLOSE trip_cursor;
END;
```

curseur - FETCH

```
DECLARE
    CURSOR trip_cursor IS
        SELECT bt_id_pk, bt_hotel_id
        FROM business_trips;

    trip_record trip_cursor%ROWTYPE
BEGIN
    OPEN trip_cursor;
    LOOP
        FETCH trip_cursor INTO trip_record;
        EXIT WHEN trip_cursor%NOTFOUND;
        INSERT INTO copy_of_business_trips (bt_id_pk, bt_hotel_id)
        VALUES (trip_record.bt_id_pk, trip_record.bt_hotel_id);
    END LOOP;
    CLOSE trip_cursor;
END;
```

retour sur la gestion d'erreur

Il est possible d'attraper, gérer plusieurs erreurs dans un programme. Vous n'avez qu'à les lister les cas et les gérer.

```
CREATE PROCEDURE p_A
AS
    CURSOR c2 (inLangue IN number) IS
        SELECT pClient, cClient
        FROM Client
        WHERE pLangue = inLangue;
BEGIN

    ... pl/sql code ...

EXCEPTION
    WHEN NOT_FOUND THEN
        DBMS_OUTPUT.PUT_LINE(sqlerrm);
    WHEN OTHERS THEN
        NULL;
END p_A;
/
```

création d'une trappe

```
CREATE PROCEDURE p_A
AS
    table_does_not_exist      EXCEPTION;
    PRAGMA EXCEPTION_INIT (table_does_not_exist, -4910);
BEGIN

    ... pl/sql code ...

EXCEPTION
    WHEN table_does_not_exist THEN
        RAISE;
    WHEN NOT_FOUND THEN
        DBMS_OUTPUT.PUT_LINE(sqlerrm);
        RAISE;
    WHEN OTHERS THEN
        NULL;
END p_A;
/
```


Le pl/sql est un langage complet qui permet de faire tout ce qui est nécessaire afin de gérer de l'information et des données. Voici deux notions très importantes concernant les bases de données

- ▶ consistance;
- ▶ transaction;
- ▶ problématique de l'aller-retour. (*roundtrip*)

- ▶ En aucun temps, une base de données ne peut être, se retrouver, dans un état incertain. La consistance des données et le statut de l'information doivent toujours être valides.
- ▶ Une BD, des tables, un schéma mal modélisé auront certainement un impact sur la redondance, et l'unicité de l'information. Par répercussion, il y aura aussi un impact sur la performance et/ou la constance (dans le sens de déterministe \equiv toujours donner le même résultat).

- ▶ Le mode transactionnel existe afin de gérer les multiples transactions (INSERT, UPDATE, DELETE) souvent simultanées pour que la BD reste consistante.
- ▶ Si une base de données est consistante avant le début d'une transaction elle devra OBLIGATOIREMENT l'être après.
- ▶ C'est tout ou rien.
- ▶ Une transaction est composée de: 1 DELETE 1 UPDATE 1 INSERT, impossible que seulement le DELETE soit traité.
- ▶ Les transactions non terminées ne sont pas visibles par les autres utilisateurs ou sessions actives.

- ▶ explications, et exemples sur le roundtrip.

- ▶ Le pl/sql est mature et complet, il est aussi très performant.
- ▶ De plus il évite plusieurs problèmes.
- ▶ Les programmes sont compilés dans le gestionnaire BD, le SGBDR.
- ▶ Il est aussi possible de programmer en C dans Oracle. Ce langage se nomme PRO*C.
- ▶ Il est possible de faire des programmes Java et de les rendre disponibles dans Oracle par l'entremise de *packages*.
- ▶ Dans SQL Server de Microsoft Transact-SQL ou TSQL est un équivalent à pl/sql d'Oracle.