PLSQL

BDTN

2

Interaction avec la base oracle : curseurs explicites (suite)

- FOR UPDATE
- Avec paramètres

Gestion des exceptions

Curseurs explicites (1)

SELECT retourne plus d’une ligne

OPEN – LOOP - CLOSE

Intérêt des structures – déclaration avec %ROWTYPE

Boucle FOR pour les curseurs

Curseur FOR UPDATE

Que se passe-t-il si un autre utilisateur met à jour les lignes de la table SHOW_SHW après que le curseur soit ouvert ?

Et si vous vouliez également mettre à jour ces lignes ?

Curseur FOR UPDATE

Transaction 1

BEGIN

-- 1 afficher les réservations du clients 1

prc_list_bkg(1);

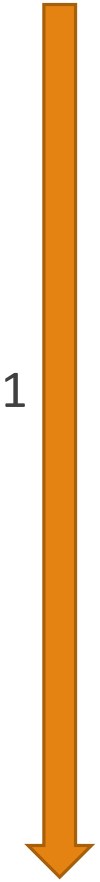
-- 3 MAJ de certaines réservations du client 1

prc_update_seat_bkg(1, 1);

-- 4 Fin de transaction 1 validation

COMMIT;

END;



Transaction 2

BEGIN

-- 2 afficher les réservations du client 1

prc_list_bkg(1);

-- 5 MAJ de certaines réservations du client 1

prc_update_seat_bkg(1, 2);

-- 6 Fin de transaction 2 validation

COMMIT;

END;

Cursor FOR UPDATE

Lignes verrouillées à l'ouverture du curseur

Tant que le curseur est ouvert, les autres utilisateurs ne peuvent pas modifier les lignes

Avec WHERE CURRENT OF

- Update
- Delete

- NOWAIT : provoquera une erreur oracle si les lignes sont déjà verrouillées
- WAIT n : attente de n secondes
- Ni l'un ni l'autre : attendra indéfiniment jusqu'à ce que les lignes soient disponibles

Cursor avec FOR UPDATE

```
DECLARE
  CURSOR cur_hpr IS
    SELECT *
      FROM has_price_hpr
      FOR UPDATE NOWAIT;
BEGIN
  FOR v_hpr_record IN cur_hpr LOOP
    DBMS_OUTPUT.PUT_LINE(v_hpr_record.hpr_shw_id || ' '
                                        || v_hpr_record.hpr_seat_price);
    UPDATE has_price_hpr
      SET hpr_seat_price = hpr_seat_price * 1.05
      WHERE CURRENT OF cur_hpr;
  END loop;
END;
```

```
DECLARE
  CURSOR cur_hpr IS
    SELECT *
      FROM has_price_hpr
      INNER JOIN show_shw
      ON shw_id = hpr_shw_id
      FOR UPDATE OF hpr_seat_price;
BEGIN
  FOR v_hpr_record IN cur_hpr LOOP
    DBMS_OUTPUT.PUT_LINE(v_hpr_record.shw_title || ' '
                         || v_hpr_record.hpr_seat_price);
    UPDATE has_price_hpr
      SET hpr_seat_price = hpr_seat_price * 1.05
      WHERE CURRENT OF cur_hpr;
  END loop;
END;
```

Curseur FOR UPDATE

Transaction 1

BEGIN

-- 1 afficher les réservations du clients 1 – FOR UPDATE

prc_list_bkg(1);

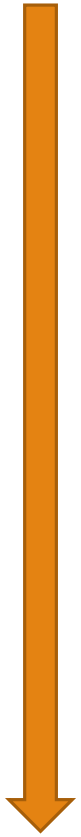
-- 2 MAJ de certaines réservations du client 1

prc_update_seat_bkg(1, 1);

-- 3 Fin de transaction 1 validation

COMMIT;

END;



Transaction 2

BEGIN

-- 4 afficher les réservations du client 1 FOR UPDATE

prc_list_bkg(1);

-- 5 MAJ de certaines réservations du client 1

prc_update_seat_bkg(1, 2);

-- 6 Fin de transaction 2 validation

COMMIT;

END;

Curseurs avec paramètres

Curseur utilisé plusieurs fois avec des valeurs différentes pour la clause WHERE

Curseurs imbriqués

```
DECLARE
   CURSOR cur_shw (p_month NUMBER) IS
      SELECT *
        FROM show_shw
        WHERE EXTRACT (MONTH FROM shw_date) = p_month;
   v_month NUMBER;

BEGIN
   SELECT MIN(EXTRACT (MONTH FROM shw_date))
     INTO v_month
     FROM show_shw;
   FOR v_shw_record IN cur_shw(v_month) LOOP
     DBMS_OUTPUT.PUT_LINE(v_shw_record.shw_title || ' '
                          || TO_CHAR(v_shw_record.shw_date, 'DD Month'));
   END loop;

END;
```

Gestion des exceptions

Position dans le bloc PL/SQL

Types d’erreurs – types d’exceptions

Supports Oracle : Chapitre 7, Chapitre 9 - section 4

Erreur oracle

```
DECLARE
v_tsh_name type_show_tsh.tsh_name%TYPE;

BEGIN
  SELECT tsh_name
    INTO v_tsh_name
    FROM type_show_tsh
    WHERE tsh_id = 200;

  DBMS_OUTPUT.PUT_LINE(v_tsh_name)
END;
```

```
Rapport d'erreur -
ORA-01403: no data found
ORA-06512: at line 4
01403. 00000 -  "no data found"
*Cause:    No data was found from the
objects.
*Action:   There was no data from the
objects which may be due to end of
fetch.
```

```
BEGIN
  SELECT tsh_name
      INTO v_tsh_name
       FROM type_show_tsh
      WHERE tsh_id = 200;

 DBMS_OUTPUT.PUT_LINE(v_tsh_name);

EXCEPTION
WHEN NO_DATA_FOUND THEN
  DBMS_OUTPUT.PUT_LINE('No such type of show');

WHEN OTHERS THEN
  DBMS_OUTPUT.PUT_LINE('Error occurred');
END;
```

Syntaxe

```
EXCEPTION
WHEN exception1 [OR exception2 . . .] THEN
statement1;
statement2;
. . .
[WHEN exception3 [OR exception4 . . .] THEN
statement1;
statement2;
. . .]
[WHEN OTHERS THEN
statement1;
statement2;
. . .]
```

Exceptions / Gestion des erreurs

Erreurs

- Erreurs déclenchées par oracle
- Erreurs définies par le programmeur

2 possibilités

- Traitée dans la section exception
- Le bloc se termine par une erreur et l'exception est propagée (Oracle section 7_4)

Catégories d'exceptions

- Erreurs se déclenchant implicitement
  - Erreurs prédéfinies
  - Erreurs Oracle non prédéfinies
- Erreurs se déclenchant explicitement

Erreurs prédéfinies

Nommées

- ZERO_DIVIDE
- NO_DATA_FOUND
  - SELECT … INTO ne retourne aucune ligne
- TOO_MANY_ROWS
  - SELECT … INTO concerne plus d'une ligne
- DUP_VAL_ON_INDEX
  - Doublon sur une colonne « UNIQUE »
- Etc.
- Liste des erreurs prédéfinies – PL/SQL user's guide and reference
  - https://docs.oracle.com/cd/A97630_01/appdev.920/a96624/07_errs.htm#784

Erreurs oracle non nommées

Peut être récupérée par WHEN OTHERS

Peut être nommée : EXCEPTION_INIT

Faire apparaître la nature de l'erreur

- SQLCODE
  - Retourne le code de la dernière erreur
  - A utiliser uniquement dans la section exception
  - sql_err := SQLCODE;
- SQLERRM
  - Retourne le message d'erreur de la dernière erreur
  - sql_msg := SQLERRM;

Erreurs oracle non nommées

INSERT INTO type_show_tsh

VALUES (7, NULL);

Erreur commençant à la ligne: 35 de la commande -

INSERT INTO type_show_tsh

VALUES (7, NULL)

Rapport d'erreur -

Erreur SQL : ORA-01400: cannot insert NULL into ("BDD1"."TYPE_SHOW_TSH"."TSH_NAME")

01400. 00000 - "cannot insert NULL into (%s)"

*Cause: An attempt was made to insert NULL into previously listed objects.

*Action: These objects cannot accept NULL values.

```
DECLARE
e_insert EXCEPTION;
PRAGMA EXCEPTION_INIT(e_insert, -01400);
v_error_code NUMBER;
v_error_msg VARCHAR2(255);

BEGIN
INSERT INTO type_show_tsh
VALUES (7, NULL);

EXCEPTION
WHEN e_insert THEN
  DBMS_OUTPUT.PUT_LINE('Error on INSERT');
  v_error_code := SQLCODE;
  v_error_msg := SQLERRM;
  INSERT INTO error_log(e_user, e_date, error_code, error_msg)
  VALUES(USER, SYSDATE, v_error_code, v_error_message);
END;
```

Déclarer l'exception

Associer numéro de l'exception oracle et exception déclarée

Gérer l'exception

Insérer code d'erreur et message dans une table de log

Erreurs définies par le programmeur

Les salaires ne peuvent pas être diminués

Les notes ne peuvent pas être mises à jour le dimanche

Une erreur de saisie ne provoque pas la mise à jour attendue

Une instruction UPDATE ou DELETE ne modifie/supprime aucune ligne

```
CREATE OR REPLACE PROCEDURE del_bkg_noexcep(p_bkg_id IN booking_bkg.bkg_id%TYPE)
AS



BEGIN
  DELETE FROM booking_bkg
    WHERE bkg_id = p_bkg_id;

  dbms_output.put_line('Booking num : ' || p_bkg_id || ' has been successfully removed');

END del_bkg_noexcep ;
```

```
SELECT * FROM booking_bkg;
```

	BKG_ID	BKG_DATE	BKG_TOTAL_SEAT	BKG_CST_ID	BKG_SHW_ID	BKG_TPR_ID
1	100	29/09/12	2	1	10	1
2	101	29/09/12	4	3	10	2
3	102	28/08/12	1	2	14	1

```
BEGIN
del_bkg_noexcep(100);
del_bkg_noexcep(10);
del_bkg_noexcep(101);
END;
```

```
Booking num : 100 has been successfully removed
Booking num : 10 has been successfully removed
Booking num : 101 has been successfully removed
```

```
SELECT * FROM booking_bkg;
```

	BKG_ID	BKG_DATE	BKG_TOTAL_SEAT	BKG_CST_ID	BKG_SHW_ID	BKG_TPR_ID
1	102	28/08/12	1	2	14	1

```
CREATE OR REPLACE PROCEDURE del_bkg(p_bkg_id IN booking_bkg.bkg_id%TYPE) AS

  e_bkg_id_entry EXCEPTION;

BEGIN
  DELETE FROM booking_bkg
    WHERE bkg_id = p_bkg_id;

  IF (SQL%NOTFOUND) THEN
    RAISE e_bkg_id_entry;
  END IF;

  DBMS_OUTPUT.PUT_LINE('Booking num : ' || p_bkg_id || ' has been successfully removed');

EXCEPTION
WHEN e_bkg_id_entry THEN
  DBMS_OUTPUT.PUT_LINE('no such booking ' || p_bkg_id);
END del_bkg ;
```

SELECT * FROM booking_bkg;

	BKG_ID	BKG_DATE	BKG_TOTAL_SEAT	BKG_CST_ID	BKG_SHW_ID	BKG_TPR_ID
1	100	29/09/12	2	1	10	1
2	101	29/09/12	4	3	10	2
3	102	28/08/12	1	2	14	1

```
BEGIN
del_bkg(100);
del_bkg(10);
del_bkg(101);
END;
```

```
Booking num : 100 has been successfully removed
no such booking 10
Booking num : 101 has been successfully removed
```

SELECT * FROM booking_bkg;

	BKG_ID	BKG_DATE	BKG_TOTAL_SEAT	BKG_CST_ID	BKG_SHW_ID	BKG_TPR_ID
1	102	28/08/12	1	2	14	1

Bonnes pratiques

Toujours penser à la section exception

Nommer les exceptions plutôt que tout traiter avec « OTHER »

Décider si la partie exception doit exécuter commit, rollback, ou laisser la transaction se poursuivre

Voir également : PLSQL 7_1 p23-24