

PL/SQL

BDTN 2018

1

PL/SQL vs SQL

Variables

Blocs – autonomes – procédures – fonctions

Interaction avec la base oracle – curseurs implicites

Structures de contrôles : IF

PL/SQL et SQL

SQL

- Standard ANSI (American National Standards Institute)
- Non procédural

PL/SQL

- Oracle
- Extension procédurale
- Utiliser SQL

Quelques-uns des bénéfices de PL/SQL

- Meilleures performances
- Portabilité

Blocs PL/SQL

[Oracle - iLearning - PLSQL 1 3](#)

T5 à 14

Déclaration

```
DECLARE
  v_mo_length NUMBER;
  v_star_addr VARCHAR2(50) DEFAULT 'Hollywood';
  v_price NUMBER(10,2) := 0;
  v_birthdate DATE := TO_DATE('25/10/2016', 'dd/mm/yyyy');
  v_picture BLOB;

  v_mo_genre movie.mo_genre%TYPE;
```

Colonnes | Données | Model | Contraintes | Droits | Statistiques | Déclen



▼ Actions...

	❖ COLUMN_NAME	❖ DATA_TYPE	❖ NULLABLE	I
1	MO_ID	NUMBER(5,0)	No	
2	MO_TITLE	VARCHAR2(60 BYTE)	No	
3	MO_YEAR	NUMBER(4,0)	Yes	
4	MO_GENRE	VARCHAR2(25 BYTE)	Yes	
5	MO_LENGTH	NUMBER(3,0)	Yes	
6	MO_STUDIO	NUMBER(3,0)	Yes	
7	MO_DIRECTOR	NUMBER(3,0)	Yes	
8	MO_ORIGINAL	NUMBER(5,0)	Yes	

Attribut %TYPE

Evite certaines erreurs

- Erreurs de type, précision

Changement de type au niveau de la colonne

- Le code n'est pas à reprendre

Affectation et conversions de type

Conversions implicites

Conversions explicites

- TO_CHAR(value, fmt)
- TO_NUMBER(value, fmt)
- TO_DATE(value, fmt)


```
-- Activation of the DBMS_OUTPUT package  
-- Has to be done once
```

```
SET SERVEROUTPUT ON
```

```
DECLARE
```

```
  v_birthdate DATE;
```

```
BEGIN
```

```
v_birthdate := TO_DATE('25/10/2016', 'DD/MM/YYYY');
```

```
DBMS_OUTPUT.PUT_LINE(TO_CHAR(v_birthdate, 'DD Month YYYY'));
```

```
END;
```



Sortie de script x



Tâche terminée en 0,015 secondes

Procédure PL/SQL terminée.

25 Octobre 2016

Conventions de nommage

ilearning.oracle.com PLSQL 2 7

T 9 à 11

SQL et PL/SQL

Peuvent être utilisées

- SELECT
- Instructions de modification de données (DML)
- Instructions de contrôle des transactions (TCL)

Ne peuvent pas être utilisées directement

- Instructions de définition de données (DDL)
- Instructions de contrôle de données (DCL)

Récupérer une ligne – SELECT ... INTO

```
DECLARE
  v_title movie.mo_title%TYPE;
  v_year movie.mo_year%TYPE;

BEGIN
  SELECT mo_title, mo_year
    INTO v_title, v_year
   FROM movie
  WHERE mo_id = 1;
  DBMS_OUTPUT.PUT_LINE('Movie : ' || v_title || ' ' || v_year);
END;
```

Sortie de script x

Tâche terminée en 0,004 secondes

Procédure PL/SQL terminée.

Movie : 12 angry men 1957

&movie_id

```
DECLARE
  v_title movie.mo_title%TYPE;
  v_year movie.mo_year%TYPE;
  v_mo_id movie.mo_id%TYPE := &movie_id;

BEGIN
  SELECT mo_title, mo_year
    INTO v_title, v_year
   FROM movie
  WHERE mo_id = v_mo_id;
  DBMS_OUTPUT.PUT_LINE('Movie : ' || v_title);
END;
```

Entrer une variable de substitution

Entrez la valeur de movie_id ::

OK Annuler

```

DECLARE
v_title movie.mo_title%TYPE;
v_year movie.mo_year%TYPE;

BEGIN
  SELECT mo_title, mo_year
    INTO v_title, v_year
   FROM movie;
  DBMS_OUTPUT.PUT_LINE('Movie : ' || v_title || ' ' || v_year);
END;

WHERE mo_id = 1;

```

Sortie de script x

Tâche terminée en 0,017 secondes

```

BEGIN
  SELECT mo_title, mo_year
    INTO v_title, v_year
   FROM movie;
  DBMS_OUTPUT.PUT_LINE('Movie : ' || v_title || ' ' || v_year);
END;
Rapport d'erreur -
ORA-01422: exact fetch returns more than requested number of rows
ORA-06512: at line 6
01422. 00000 - "exact fetch returns more than requested number of rows"

```



Curseur implicite

Un curseur est :

- Un pointeur vers une zone mémoire dans laquelle Oracle stocke les données traitées par une instructions SQL
- Un nom pour cette zone mémoire

Les curseurs implicites sont définis automatiquement

- Pour toutes les instructions LMD (INSERT, UPDATE, DELETE, MERGE)
- Pour les instructions SELECT (une ligne)

Ils sont désignés par « SQL »

Attributs des curseurs

%ROWCOUNT

- Retourne le nombre de lignes traitées

%FOUND

- TRUE si INSERT, UPDATE, DELETE, MERGE ont traité au moins une ligne

%NOTFOUND

- L'opposé de %FOUND

SQL%ROWCOUNT


```
DECLARE
v_mo_id movie.mo_id%TYPE := &movie_id;

BEGIN
DELETE FROM movie
WHERE mo_id = v_mo_id;
DBMS_OUTPUT.PUT_LINE(TO_CHAR(SQL%ROWCOUNT) || ' rows deleted');
END;
```

Sortie de script x



Tâche terminée en 0,154 secondes

END;

nouveau :DECLARE

```
v_mo_id movie.mo_id%TYPE := 1;
```

BEGIN

```
DELETE FROM movie
```

```
WHERE mo_id = v_mo_id;
```

```
DBMS_OUTPUT.PUT_LINE(TO_CHAR(SQL%ROWCOUNT) || ' rows deleted');
```

END;

Procédure PL/SQL terminée.

1 rows deleted

```
DECLARE
    v_mo_id movie.mo_id%TYPE := &movie_id;

BEGIN
    DELETE FROM movie
    WHERE mo_id = v_mo_id;
    IF (SQL%NOTFOUND) THEN
        DBMS_OUTPUT.PUT_LINE('no row');
    ELSE
        DBMS_OUTPUT.PUT_LINE(TO_CHAR(SQL%ROWCOUNT) || ' rows deleted');
    END IF;
END;
```

Sortie de script x



Tâche terminée en 0,007 secondes

END;

Procédure PL/SQL terminée.

no row

Contrôle des transactions

[Oracle iLearning PLSQL 3.4](#)

T 5 à 13

Sous-programmes

Nommés

Sont compilés une fois et sont stockés

Peuvent être réutilisés, partagés

Placés dans des packages

Procédures

```
CREATE [OR REPLACE] PROCEDURE procedure_name (Mandatory)
[(parameter1 [mode] datatype1, (Optional)
parameter2 [mode] datatype2, ...)]
IS|AS (Mandatory)
[local_variable_declarations; ...] (Optional)

BEGIN (Mandatory)
    SQL and PL/SQL statements;
    EXCEPTION (Optional)
        WHEN exception-handling actions;
END [name]; (Mandatory)
```

```
CREATE OR REPLACE PROCEDURE increase_price ( p_coeff IN NUMBER) IS
```

```
  v_nb_rows NUMBER(4);
```

```
BEGIN
```

```
  UPDATE has_price_hpr
```

```
    SET hpr_seat_price = hpr_seat_price * p_coeff
```

```
    WHERE hpr_shw_id NOT IN (SELECT bkg_shw_id FROM booking_bkg);
```

```
  v_nb_rows := SQL%ROWCOUNT;
```

```
  IF v_nb_rows > 0 THEN
```

```
    DBMS_OUTPUT.PUT_LINE(v_nb_rows || ' rows updated');
```

```
  ELSE
```

```
    DBMS_OUTPUT.PUT_LINE('No show without booking');
```

```
  END IF;
```

```
END increase_price;
```

```
/
```



```
BEGIN
```

```
  increase_price(1.2);
```

```
END;
```

Paramètre :

IN mode

pas de taille

Variable locale

Fin de procédure

Appel depuis un bloc
anonyme

Connexions

Page de début x Week1.sql x Week2.sql x local1.sql x Sans titre2.sql x Week 3 - /

Feuille de calcul SQL Historique

Feuille de calcul Query Builder

```
COMMIT;  
DBMS_OUTPUT.PUT_LINE('Changes committed');  
DBMS_OUTPUT.PUT_LINE(v_nb_rows || ' rows updated');  
ELSE  
    DBMS_OUTPUT.PUT_LINE('No show without booking');  
END IF;  
END;  
  
BEGIN  
    increase_price(1.2);  
END;
```

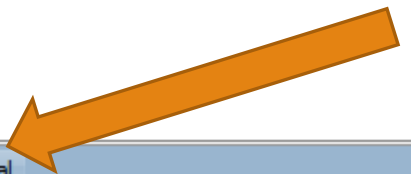
Sortie de script x

Tâche terminée en 0,094 secondes

Elément Procedure INCREASE_PRICE compilé

Errors: check compiler log

Compilateur - Journal



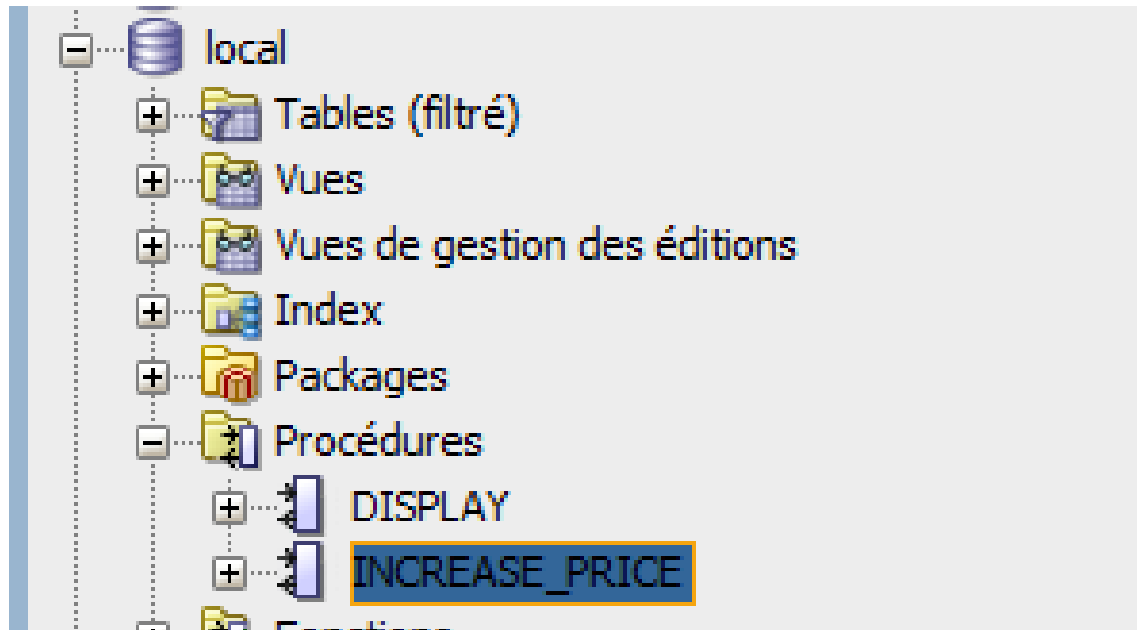
```
6      SET hpr_seat_price = hpr_seat_price * p_coeff
7      WHERE hpr_shw_id NOT IN (SELECT bkg_shw_id FROM booki
8 v_nb_rows := SQL%rowcount;
9 IF v_nb_rows > 0
10     COMMIT;
11     DBMS_OUTPUT.PUT_LINE('Changes committed');
12     DBMS_OUTPUT.PUT_LINE(v_nb_rows || ' rows updated');
```

Compilateur - Journal

Projet : C:\Users\roueche\AppData\Roaming\SQL Developer\system4.1.2.20.64\p.sqldeveloper.12.2.0.20.64\

Procédure BDD1.INCREASE_PRICE@local

Erreur(10,5): PLS-00103: Encountered the symbol "COMMIT" when expecting one of the following: *



Sélectionner la procédure
Double-clic pour ouvrir l'éditeur de
sous-programmes



```
create or replace PROCEDURE increase_price ( p_coeff IN NUMBER) IS
    v_nb_rows NUMBER;

BEGIN
    UPDATE has_price_hpr
        SET hpr_seat_price = hpr_seat_price * p_coeff
        WHERE hpr_shw_id NOT IN (SELECT bkg_shw_id FROM booking_bkg);
    v_nb_rows := SQL%rowcount;
    IF v_nb_rows > 0
        COMMIT;
        DBMS_OUTPUT.PUT_LINE('Changes committed');
        DBMS_OUTPUT.PUT_LINE(v_nb_rows || ' rows updated');
    ELSE
        DBMS_OUTPUT.PUT_LINE('No show without booking');
    END IF;
END;
```

Editeur ouvert :

- **Editer**
- **Compiler**
- **Tester**
- **Déboguer**

Fonctions

```
CREATE [OR REPLACE] FUNCTION function_name [(parameter1 [mode1] datatype1, ...)]  
RETURN datatype  
IS|AS  
  [local_variable_declarations; ...]  
BEGIN  
  -- actions;  
  RETURN expression;  
END [function_name];
```

```
CREATE OR REPLACE FUNCTION booking_cost(p_bkg_id IN booking_bkg.bkg_id%TYPE)  
RETURN NUMBER IS
```

```
v_cost NUMBER;
```

```
BEGIN
```

```
SELECT hpr_seat_price * bkg_total_seat  
  INTO v_cost  
  FROM booking_bkg INNER JOIN type_price_tpr  
    ON bkg_tpr_id = tpr_id  
    INNER JOIN has_price_hpr  
    ON hpr_tpr_id = tpr_id AND bkg_shw_id = hpr_shw_id  
 WHERE bkg_id = p_bkg_id;
```

```
RETURN v_cost;  
END booking_cost;
```

Connexion à la base de données local.

ORA-01403: no data found

ORA-06512: at "BDD1.BOOKING_COST", line 4

ORA-06512: at line 7

Processus fermé.

Déconnexion de la base de données local.

Curseurs explicites

SELECT retourne plus d'une ligne

```
DECLARE
CURSOR cur_shw IS
    SELECT shw_id, shw_title, shw_date
    FROM show_shw
    WHERE EXTRACT (MONTH FROM shw_date) = 10;
v_shw_id show_shw.shw_id%TYPE;
v_shw_title show_shw.shw_title%TYPE;
v_shw_date show_shw.shw_date%TYPE;

BEGIN
OPEN cur_shw;
LOOP
    FETCH cur_shw INTO v_shw_id, v_shw_title, v_shw_date;
    EXIT WHEN cur_shw%NOTFOUND;
    DBMS_OUTPUT.PUT_LINE(v_shw_title || ' ' || TO_CHAR(v_shw_date, 'DD Month'));
END LOOP;

CLOSE cur_shw;
END;
```

```
DECLARE
```

```
CURSOR cur_shw IS
```

```
SELECT *
```

```
FROM show_shw
```

```
WHERE EXTRACT (MONTH FROM shw_date) = 10;
```

```
v_shw_record cur_shw%ROWTYPE;
```

```
BEGIN
```

```
OPEN cur_shw;
```

```
LOOP
```

```
FETCH cur_shw INTO v_shw_record;
```

```
EXIT WHEN cur_shw%NOTFOUND;
```

```
DBMS_OUTPUT.PUT_LINE(v_shw_record.shw title || ' '  
|| TO_CHAR(v_shw_record.shw_date, 'DD Month'));
```

```
END loop;
```

```
CLOSE cur_shw;
```

```
END;
```


Attributs des Curseurs

%ISOPEN

- TRUE si le curseur est ouvert

%ROWCOUNT

- Retourne le nombre de lignes récupérées jusqu'à là

%FOUND

- TRUE si le dernier FETCH a récupéré une ligne

%NOTFOUND

- Le contraire de %FOUND
- TRUE si le dernier FETCH n'a pas retourné une ligne

Plus simplement

DECLARE

CURSOR cur_shw IS

SELECT *

FROM show_shw

WHERE EXTRACT (MONTH FROM shw_date) = 10;

BEGIN

FOR v_shw_record **IN** cur_shw **LOOP**

DBMS_OUTPUT.PUT_LINE(v_shw_record.shw_title || ' ' || TO_CHAR(v_shw_record.shw_date, 'DD Month'));

END LOOP;

END;

```
CREATE OR REPLACE PROCEDURE prc_list_show(p_month IN NUMBER) IS
    CURSOR cur_shw IS
        SELECT *
            FROM show_shw
           WHERE EXTRACT (MONTH FROM shw_date) = p_month;
BEGIN
    FOR v_shw_record IN cur_shw LOOP
        DBMS_OUTPUT.PUT_LINE(v_shw_record.shw_title || ' '
                               || TO_CHAR(v_shw_record.shw_date, 'DD Month'));
    END loop;
END;
```

Dictionnaire de données

USER

- USER_TABLES
- USER_CONSTRAINTS
- USER_INDEXES
- ...

ALL

- ALL_TABLES..
- Uniquement sur les objets pour lesquels vous avez un privilège

Dictionnaire de données

Accéder à la description de la table

- DESCRIBE
- DESC USER_TABLES

Interroger le dictionnaire de données

- `SELECT table_name FROM user_tables;`