

## CASE

---

```
DECLARE
    couleur_drapeau VARCHAR(30) := 'VERT';
BEGIN
    CASE couleur_drapeau
        WHEN 'ROUGE' THEN
            DBMS_OUTPUT.PUT_LINE('Baignade interdite !!');
        WHEN 'ORANGE' THEN
            DBMS_OUTPUT.PUT_LINE('Attention, la mer est dangereuse');
        WHEN 'VERT' THEN
            DBMS_OUTPUT.PUT_LINE('Tous à l'eau !!');
        WHEN 'NOIR' THEN
            DBMS_OUTPUT.PUT_LINE('Marée noire');
        ELSE
            DBMS_OUTPUT.PUT_LINE('Drapeau non repertorié');
    END CASE;
END;
```

```
DECLARE
    couleur_drapeau VARCHAR(30) := 'VERT';
BEGIN
    CASE
        WHEN couleur_drapeau = 'VERT' THEN
            DBMS_OUTPUT.PUT_LINE('Le drapeau est vert.');
```

WHEN couleur\_drapeau = 'ORANGE' THEN  
DBMS\_OUTPUT.PUT\_LINE('Le drapeau est orange');

WHEN couleur\_drapeau = 'ROUGE' THEN  
DBMS\_OUTPUT.PUT\_LINE('Le drapeau est rouge.');

ELSE  
DBMS\_OUTPUT.PUT\_LINE('Je ne connais pas cette couleur.');

```
    END CASE;
END;
```

## LOOP

---

```
DECLARE
    index_loop INTEGER := 0;
BEGIN
    LOOP
        IF index_loop = 3 THEN
            EXIT;
        ELSE
            DBMS_OUTPUT.PUT_LINE(index_loop);
        END IF;
        index_loop := index_loop + 1;
    END LOOP;
END;
```

## SUBTYPE

---

```
SET SERVEROUTPUT ON;

DECLARE
    SUBTYPE salaire IS NUMBER(4);
    SUBTYPE date_nn IS DATE NOT NULL;

    premiere_annee salaire := 2000;
    premier_jour date_nn := SYSDATE;

BEGIN

    DBMS_OUTPUT.PUT_LINE(premiere_annee);
    DBMS_OUTPUT.PUT_LINE(premier_jour);

END;
```

## %TYPE

---

```
SET SERVEROUTPUT ON;

DECLARE

    continent regions.region_name%TYPE;

BEGIN

    continent := 'Europe';
    DBMS_OUTPUT.PUT_LINE(continent);

END;
```

## RECORD

---

```
SET SERVEROUTPUT ON;

DECLARE

    TYPE fiche_parking IS RECORD
    (
        prenom VARCHAR2(50),
        nom VARCHAR2(50) NOT NULL := 'XXX',
        nombre NUMBER NOT NULL DEFAULT 1
    );

    fiche1 fiche_parking;

BEGIN

    fiche1.prenom := 'Anthony';
    fiche1.nom := 'Cosson';

    DBMS_OUTPUT.PUT_LINE('Prenom : ' || fiche1.prenom);
    DBMS_OUTPUT.PUT_LINE('Nom : ' || fiche1.nom);
    DBMS_OUTPUT.PUT_LINE('Nombre voiture : ' || fiche1.nombre);

END;
```

**%ROWTYPE** \_\_\_\_\_

```
SET SERVEROUTPUT ON;
```

```
DECLARE
```

```
    region_rec regions%ROWTYPE;
```

```
BEGIN
```

```
    region_rec.region_id = 5;
```

```
    region_rec.region_name = 'Antarctique';
```

```
    DBMS_OUTPUT.PUT_LINE('Id de la region : ' || region_rec.region_id);
```

```
    DBMS_OUTPUT.PUT_LINE('Nom de la region : ' || region_rec.region_name);
```

```
END;
```

**COLLECTION INDEX BY TABLE** \_\_\_\_\_

```
SET SERVEROUTPUT ON;
```

```
DECLARE
```

```
    TYPE population IS TABLE OF NUMBER INDEX BY VARCHAR(50);
```

```
    population_ville population;
```

```
    ville VARCHAR2(50);
```

```
BEGIN
```

```
    population_ville('Soliers') := 2151;
```

```
    population_ville('Caen') := 106538;
```

```
    population_ville('Limoges') := 134577;
```

```
    population_ville('Rennes') := 213454;
```

```
    ville := population_ville.FIRST;
```

```
    WHILE ville IS NOT NULL LOOP
```

```
        DBMS_Output.PUT_LINE('La population de ' || ville
```

```
        || ' est de ' || TO_CHAR(population_ville(ville)) || ' habitants');
```

```
        ville := population_ville.NEXT(ville);
```

```
    END LOOP;
```

```
END;
```

## COLLECTION NESTED TABLE

---

```
SET SERVEROUTPUT ON;

DECLARE

    TYPE tableau_noms_objets IS TABLE OF VARCHAR2(50);

    tableau_musique tableau_noms_objets;

    objet_courant VARCHAR2(50);
BEGIN

    tableau_musique := tableau_noms_objets('Guitare','Tambour','Batterie','Basse');

    tableau_musique.DELETE(2);

    DBMS_OUTPUT.PUT_LINE(tableau_musique(1));
    --DBMS_OUTPUT.PUT_LINE(tableau_musique(2));
    DBMS_OUTPUT.PUT_LINE(tableau_musique(3));

END;
```

## COLLECTION VARRAY

---

```
SET SERVEROUTPUT ON;

DECLARE

    TYPE tableau_couleur IS VARRAY(5) OF VARCHAR2(50);

    france tableau_couleur := tableau_couleur('Bleu','Blanc','Rouge');
BEGIN

    DBMS_OUTPUT.PUT_LINE(france(1));
    DBMS_OUTPUT.PUT_LINE(france(2));
    DBMS_OUTPUT.PUT_LINE(france(3));

END;
```

## CURSEUR EXPLICITE 1ere METHODE

---

```
SET SERVEROUTPUT ON;

DECLARE
    CURSOR cursor_employees_it IS SELECT * FROM EMPLOYEES WHERE DEPARTMENT_ID = 60;

    employee employees%ROWTYPE;
BEGIN
    OPEN cursor_employees_it;

    LOOP
        FETCH cursor_employees_it INTO employee;
        DBMS_OUTPUT.PUT_LINE(employee.last_name);
        EXIT WHEN cursor_employees_it%NOTFOUND;
    END LOOP;

    CLOSE cursor_employees_it;
END;
```

## CURSEUR EXPLICITE 2eme METHODE

---

```
SET SERVEROUTPUT ON;

DECLARE
    CURSOR cursor_employees_it IS SELECT first_name,last_name FROM EMPLOYEES WHERE DEPARTMENT_ID = 60;

    employee employees%ROWTYPE;
BEGIN
    FOR employee IN cursor_employees_it LOOP
        DBMS_OUTPUT.PUT_LINE(employee.first_name);
    END LOOP;

END;
```

## VERROU FOR UPDATE

---

```
SET SERVEROUTPUT ON;

DECLARE
    CURSOR cursor_employees_it IS SELECT * FROM EMPLOYEES WHERE DEPARTMENT_ID = 60 FOR UPDATE;
BEGIN
    FOR employee IN cursor_employees_it LOOP
        UPDATE employees SET last_name = UPPER(last_name) WHERE employee_id = employee.employee_id;
        DBMS_OUTPUT.PUT_LINE(employee.first_name || ' ' || employee.last_name);
    END LOOP;

    COMMIT;
END;
```

## VERROU FOR UPDATE OF

---

```
SET SERVEROUTPUT ON;

DECLARE
    CURSOR cursor_employees_it IS SELECT * FROM EMPLOYEES WHERE DEPARTMENT_ID = 60 FOR UPDATE OF last_name;
BEGIN
    FOR employee IN cursor_employees_it LOOP
        UPDATE employees SET last_name = UPPER(last_name) WHERE employee_id = employee.employee_id;
        DBMS_OUTPUT.PUT_LINE(employee.first_name || ' ' || employee.last_name);
    END LOOP;

    COMMIT;
END;
```

## WHERE CURRENT OF

---

```
SET SERVEROUTPUT ON;

DECLARE
    CURSOR cursor_employees_it IS SELECT * FROM EMPLOYEES WHERE DEPARTMENT_ID = 60 FOR UPDATE OF last_name;
BEGIN
    FOR employee IN cursor_employees_it LOOP
        UPDATE employees SET last_name = UPPER(last_name) WHERE CURRENT OF cursor_employees_it;
        DBMS_OUTPUT.PUT_LINE(employee.first_name || ' ' || employee.last_name);
    END LOOP;

    COMMIT;
END;
```

## CURSEUR EXPLICITE PARAMETRABLE

---

```
SET SERVEROUTPUT ON;

DECLARE
    CURSOR cursor_employees_it(dep NUMBER) IS SELECT * FROM EMPLOYEES WHERE DEPARTMENT_ID = dep FOR UPDATE OF last_name;
BEGIN
    FOR employee IN cursor_employees_it(60) LOOP
        UPDATE employees SET last_name = UPPER(last_name) WHERE CURRENT OF cursor_employees_it;
        DBMS_OUTPUT.PUT_LINE(employee.first_name || ' ' || employee.last_name);
    END LOOP;

    COMMIT;
END;
```

## CURSEUR EXPLICITE 3<sup>ème</sup> METHODE

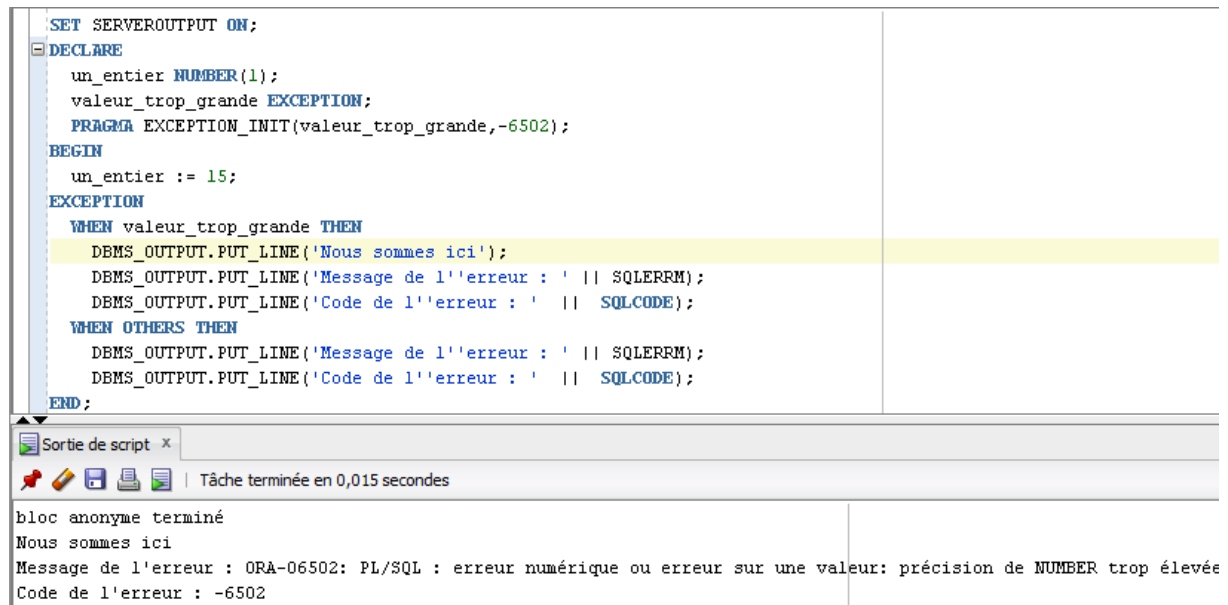
---

```
SET SERVEROUTPUT ON;

BEGIN
    FOR employee IN (SELECT first_name, last_name FROM EMPLOYEES WHERE DEPARTMENT_ID = 60) LOOP
        DBMS_OUTPUT.PUT_LINE(employee.first_name);
    END LOOP;
END;
```

## EXCEPTIONS

```
DECLARE
    --Déclaration des variables
BEGIN
    --Traitement
EXCEPTION
    --Gestion des erreurs
END;
```



The screenshot shows a SQL IDE window with a script editor and a console. The script editor contains the following code:

```
SET SERVEROUTPUT ON;
DECLARE
    un_entier NUMBER(1);
    valeur_trop_grande EXCEPTION;
    PRAGMA EXCEPTION_INIT(valeur_trop_grande,-6502);
BEGIN
    un_entier := 15;
EXCEPTION
    WHEN valeur_trop_grande THEN
        DBMS_OUTPUT.PUT_LINE('Nous sommes ici');
        DBMS_OUTPUT.PUT_LINE('Message de l''erreur : ' || SQLERRM);
        DBMS_OUTPUT.PUT_LINE('Code de l''erreur : ' || SQLCODE);
    WHEN OTHERS THEN
        DBMS_OUTPUT.PUT_LINE('Message de l''erreur : ' || SQLERRM);
        DBMS_OUTPUT.PUT_LINE('Code de l''erreur : ' || SQLCODE);
END;
```

The console output shows the following messages:

```
bloc anonyme terminé
Nous sommes ici
Message de l'erreur : ORA-06502: PL/SQL : erreur numérique ou erreur sur une valeur: précision de NUMBER trop élevée
Code de l'erreur : -6502
```

```
SET SERVEROUTPUT ON;

DECLARE
    exception_test EXCEPTION;
BEGIN
    DBMS_OUTPUT.PUT_LINE('Je suis avant le sous bloc');
    BEGIN
        DBMS_OUTPUT.PUT_LINE('Je suis avant l''erreur');
        RAISE exception_test;
        DBMS_OUTPUT.PUT_LINE('Je suis apres l''erreur');
    EXCEPTION
        WHEN exception_test THEN
            DBMS_OUTPUT.PUT_LINE('Je traite l''erreur dans le sous bloc');
    END;
    DBMS_OUTPUT.PUT_LINE('Je suis après le sous bloc');
EXCEPTION
    WHEN exception_test THEN
        DBMS_OUTPUT.PUT_LINE('Je traite l''erreur');
END;
```

## PROCEDURES (avec IN et OUT)

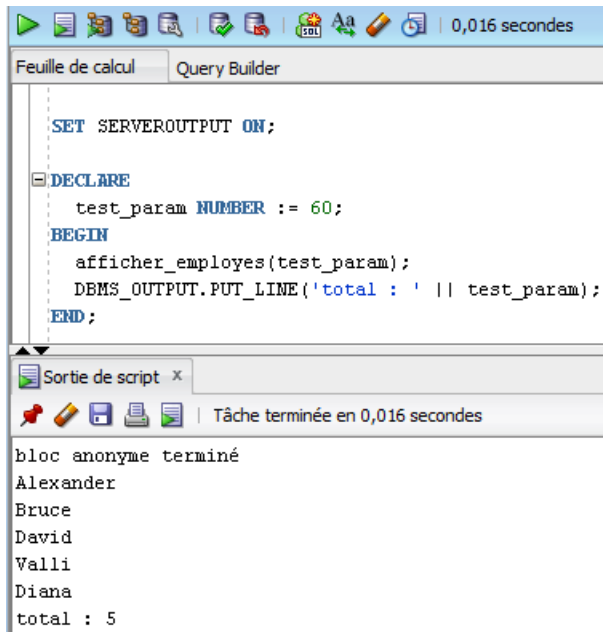
---

```
SET SERVEROUTPUT ON;

CREATE OR REPLACE PROCEDURE afficher_employes(info IN OUT NUMBER)
AS
    CURSOR cursor_employees_it IS SELECT first_name,last_name FROM EMPLOYEES WHERE DEPARTMENT_ID = info;
    employee employees%ROWTYPE;
BEGIN

    FOR employee IN cursor_employees_it LOOP
        DBMS_OUTPUT.PUT_LINE(employee.first_name);
    END LOOP;

    SELECT COUNT(*) INTO info FROM employees WHERE DEPARTMENT_ID = info;
END;
```



The screenshot displays the Oracle SQL Developer environment. The top toolbar includes icons for running, saving, and other database operations, with a timer showing 0,016 secondes. The main window is titled 'Query Builder' and contains the following PL/SQL script:

```
SET SERVEROUTPUT ON;

DECLARE
    test_param NUMBER := 60;
BEGIN
    afficher_employes(test_param);
    DBMS_OUTPUT.PUT_LINE('total : ' || test_param);
END;
```

Below the script editor, a 'Sortie de script' (Script Output) window is open, showing the execution results. It indicates that the anonymous block was completed successfully in 0,016 seconds. The output lists the first names of employees in department 60, followed by a total count.

```
bloc anonyme terminé
Alexander
Bruce
David
Valli
Diana
total : 5
```



## Les Fonctions

---

Les paramètres d'entrées sont exclusivement de type IN

- Les paramètres d'entrées doivent être de type SQL et non PL/SQL
  - Le paramètre de retour doit être de type SQL et non PL/SQL
  - Les fonctions ne doivent pas faire de DML (INSERT, UPDATE, DELETE)
- Les fonctions

```
CREATE OR REPLACE FUNCTION multiplier_par_deux(nombre_a_multiplier IN NUMBER)
RETURN NUMBER
IS
resultat NUMBER;
BEGIN
    resultat := nombre_a_multiplier * 2;
    RETURN resultat;
END;
```