Résolution TP Pratique 1

Énoncé : Bloc PL/SQL avec un enregistrement basé sur `COUNTRIES`, variable de substitution pour l'ID pays, affichage avec `DBMS_OUTPUT`. Tester avec CA, DE, UK, USA.

Solution

```
SET SERVEROUTPUT ON
SET VERIFY OFF

DEFINE p_country_id = 'CA'

DECLARE
  -- a. Enregistrement basé sur la structure de la table
COUNTRIES   rec_country countries%ROWTYPE;
  v_country_id VARCHAR2(2) := '&p_country_id';

BEGIN
  SELECT country_id, country_name, region_id
    INTO rec_country.country_id, rec_country.country_name,
        rec_country.region_id
    FROM countries
   WHERE country_id = UPPER(v_country_id);

  -- c. Affichage des informations sélectionnées
  DBMS_OUTPUT.PUT_LINE('Pays : ' || rec_country.country_name);
  DBMS_OUTPUT.PUT_LINE('ID   : ' || rec_country.country_id);
DBMS_OUTPUT.PUT_LINE('Region ID : ' || rec_country.region_id);

EXCEPTION
  WHEN NO_DATA_FOUND THEN
    DBMS_OUTPUT.PUT_LINE('Aucun pays trouve pour l''ID : ' ||
v_country_id);

END;
/
```

Tests (d. Exécuter pour CA, DE, UK, USA) :
Changer à chaque fois la ligne `DEFINE` avant d'exécuter le bloc :

- `DEFINE p_country_id = 'CA'`
- `DEFINE p_country_id = 'DE'`
- `DEFINE p_country_id = 'UK'`
- `DEFINE p_country_id = 'USA'`

Variante avec SELECT INTO sur l'enregistrement complet :

```
SELECT country_id, country_name, region_id
```

```
  INTO rec_country
  FROM countries
 WHERE country_id = UPPER(v_country_id);
```

Exercice 2 – Noms des départements avec table INDEX BY (VARCHAR2)

Énoncé : Table INDEX BY pour stocker les noms de départements, boucle avec correspondance COUNTER → DEPARTMENT_ID, puis une autre boucle pour afficher les noms.

Solution

```
SET SERVEROUTPUT ON

DECLARE
  -- a. Table INDEX BY pour stocker les noms de départements
  TYPE my_dept_table IS TABLE OF
departments.department_name%TYPE     INDEX BY
PLS_INTEGER;   v_depts my_dept_table;

  -- Table pour mapper COUNTER -> DEPARTMENT_ID (1->10,
2->20, ...)   TYPE t_id IS TABLE OF NUMBER INDEX BY
PLS_INTEGER;   v_dept_ids t_id;

  v_counter PLS_INTEGER;
BEGIN
  v_dept_ids(1) := 10;
v_dept_ids(2) := 20;
v_dept_ids(3) := 50;
v_dept_ids(4) := 60;
v_dept_ids(5) := 80;
v_dept_ids(6) := 90;
v_dept_ids(7) := 110;

  -- b. Boucle : récupérer le nom de chaque département et le
       stocker dans la table INDEX BY
  FOR v_counter IN 1..7 LOOP
    SELECT department_name
      INTO v_depts(v_counter)
      FROM departments
     WHERE department_id = v_dept_ids(v_counter);

  END LOOP;

  -- c. Autre boucle : afficher les noms à partir de la table
INDEX BY
  FOR v_counter IN 1..7 LOOP
    DBMS_OUTPUT.PUT_LINE(v_depts(v_counter));
```

```
  END LOOP; EXCEPTION
  WHEN NO_DATA_FOUND THEN
    DBMS_OUTPUT.PUT_LINE('Departement non trouve pour counter '
        || v_counter);
END;
/
```

Exercice 3 – Toutes les infos département avec table INDEX BY de RECORD

Énoncé : Table INDEX BY d'enregistrements pour stocker numéro, nom et localisation de chaque département. Même correspondance COUNTER → DEPARTMENT_ID, sortie de la boucle quand le compteur atteint 7. Puis une autre boucle pour tout afficher.

Solution

```
SET SERVEROUTPUT ON

DECLARE
  -- a. Type enregistrement pour une ligne département
(numéro, nom, localisation)   TYPE dept_rec IS RECORD (
dept_id   departments.department_id%TYPE,     dept_name
departments.department_name%TYPE,     dept_loc
departments.location_id%TYPE   );

  -- Table INDEX BY de ces enregistrements
  TYPE my_dept_table IS TABLE OF
dept_rec     INDEX BY PLS_INTEGER;
v_depts my_dept_table;

  TYPE t_id IS TABLE OF NUMBER INDEX BY PLS_INTEGER;
v_dept_ids t_id;

  v_counter
PLS_INTEGER; BEGIN
v_dept_ids(1) := 10;
v_dept_ids(2) := 20;
v_dept_ids(3) := 50;
v_dept_ids(4) := 60;
v_dept_ids(5) := 80;
v_dept_ids(6) := 90;
v_dept_ids(7) :=
110;

  -- b. Boucle : récupérer toutes les infos et les stocker ;
       sortir quand counter = 7
  v_counter := 1;
```

```
  LOOP
    SELECT department_id, department_name, location_id
      INTO v_depts(v_counter).dept_id,
        v_depts(v_counter).dept_name, v_depts(v_counter).dept_loc
      FROM departments
     WHERE department_id = v_dept_ids(v_counter);

    EXIT WHEN v_counter = 7;
v_counter := v_counter + 1;
END LOOP;

  -- c. Autre boucle : afficher les infos depuis la table INDEX
BY
  FOR v_counter IN 1..7 LOOP
    DBMS_OUTPUT.PUT_LINE(
      'Departement ' || v_depts(v_counter).dept_id ||
      ' : ' || v_depts(v_counter).dept_name ||
      ' (Location : ' || v_depts(v_counter).dept_loc || ')'
    );
  END LOOP; EXCEPTION
  WHEN NO_DATA_FOUND THEN
    DBMS_OUTPUT.PUT_LINE('Departement non trouve pour counter '
        || v_counter);

END;
/
```

Variante avec `FOR v_counter IN 1..7` pour la première boucle (équivalent « sortir à 7 ») :

```
  FOR v_counter IN 1..7 LOOP
    SELECT department_id, department_name, location_id
      INTO v_depts(v_counter).dept_id,
        v_depts(v_counter).dept_name, v_depts(v_counter).dept_loc
      FROM departments
     WHERE department_id = v_dept_ids(v_counter);
  END LOOP;
```

Résolution TP Pratique 2

1. Procédure ADD_JOB

a. Création de la procédure

```
CREATE OR REPLACE PROCEDURE add_job (
p_job_id    IN jobs.job_id%TYPE,
p_job_title IN jobs.job_title%TYPE
) IS
BEGIN
  INSERT INTO jobs (job_id, job_title)
  VALUES (p_job_id, p_job_title);
  COMMIT;
END add_job;
/
```

b. Compilation et premier appel

```
-- Invoquer avec IT_DBA et Database
Administrator EXEC add_job('IT_DBA',
'Database Administrator');

-- Vérifier le résultat
SELECT job_id, job_title, min_salary, max_salary FROM jobs WHERE
        job_id = 'IT_DBA';
```

Résultat attendu :

JOB_ID	JOB_TITLE	MIN_SALARY	MAX_SALARY
IT_DBA	Database Administrator		

c. Deuxième appel (ST_MAN, Stock Manager)

```
EXEC add_job('ST_MAN', 'Stock Manager');
```

Comportement : erreur (contrainte d'unicité ou contrainte de clé primaire).
Pourquoi ? `ST_MAN` existe déjà dans `JOBS`. Un nouvel `INSERT` avec le même `job_id` viole la clé primaire.

Pour éviter l'erreur, on peut gérer l'exception :

```
CREATE OR REPLACE PROCEDURE add_job (
p_job_id    IN jobs.job_id%TYPE,
p_job_title IN jobs.job_title%TYPE
) IS
BEGIN
  INSERT INTO jobs (job_id, job_title)
  VALUES (p_job_id, p_job_title);
```

```
  COMMIT;
  DBMS_OUTPUT.PUT_LINE('Emploi ajoute : ' || p_job_id || ' - ' ||
        p_job_title);
EXCEPTION
  WHEN DUP_VAL_ON_INDEX THEN
    DBMS_OUTPUT.PUT_LINE('Erreur : l''ID d''emploi existe deja ('
        || p_job_id || ')');
END add_job;
/
```

2. Procédure UPD_JOB

a. Création avec gestion d'exception (aucune mise à jour)

```
CREATE OR REPLACE PROCEDURE upd_job (
p_job_id    IN jobs.job_id%TYPE,
p_job_title IN jobs.job_title%TYPE
) IS
  v_rows NUMBER;
BEGIN
  UPDATE jobs
     SET job_title =

p_job_title    WHERE job_id

= p_job_id;   v_rows :=

SQL%ROWCOUNT;

  IF v_rows = 0 THEN
    RAISE_APPLICATION_ERROR(-20001, 'Aucun emploi trouve pour
        l''ID : ' || p_job_id);
  END IF;

  COMMIT;
  DBMS_OUTPUT.PUT_LINE('Emploi mis a jour : ' || p_job_id);
EXCEPTION
  WHEN OTHERS THEN
    ROLLBACK;
    RAISE;
END upd_job;
/
```

b. Appel et vérification

```
-- Modifier IT_DBA en "Data Administrator"
EXEC upd_job('IT_DBA', 'Data Administrator');
```

```
SELECT job_id, job_title, min_salary, max_salary FROM jobs WHERE
        job_id = 'IT_DBA';
```

Résultat attendu :

JOB_ID	JOB_TITLE	MIN_SALARY	MAX_SALARY
IT_DBA	Data Administrator		

Test de la gestion d'exception (emploi inexistant) :

```
EXEC upd_job('IT_WEB', 'Web Master');
```

Message attendu : `Aucun emploi trouve pour l'ID : IT_WEB` (ou erreur -20001 affichée).

---

3. Procédure DEL_JOB

a. Création avec gestion d'exception (aucune suppression)

```
CREATE OR REPLACE PROCEDURE del_job (
p_job_id IN jobs.job_id%TYPE
) IS
  v_rows NUMBER;
BEGIN
  DELETE FROM jobs WHERE job_id =

p_job_id;   v_rows := SQL%ROWCOUNT;

  IF v_rows = 0 THEN
    RAISE_APPLICATION_ERROR(-20002, 'Aucun emploi supprime. ID
        inexistant : ' || p_job_id);
  END IF;
  COMMIT;
  DBMS_OUTPUT.PUT_LINE('Emploi supprime : ' || p_job_id);
EXCEPTION
  WHEN OTHERS THEN
    ROLLBACK;
    RAISE;
END del_job;
/
```

b. Appel et vérification

```
-- Supprimer IT_DBA
EXEC del_job('IT_DBA');
```

```
-- Vérifier : plus de ligne pour IT_DBA
SELECT job_id, job_title FROM jobs WHERE job_id = 'IT_DBA';
-- no rows selected
```

Test de la gestion d'exception (emploi inexistant) :

```
EXEC del_job('IT_WEB');
```

Sortie attendue : message d'erreur contenant *Aucun emploi supprime. ID inexistant : IT_WEB* (RAISE_APPLICATION_ERROR -20002).

4. Procédure QUERY_EMP

a. Création (paramètres OUT pour salaire et job_id)

```
CREATE OR REPLACE PROCEDURE query_emp (
p_emp_id   IN
employees.employee_id%TYPE,   p_salary
OUT employees.salary%TYPE,   p_job_id
OUT employees.job_id%TYPE
) IS
BEGIN
  SELECT salary, job_id
    INTO p_salary, p_job_id
    FROM employees
   WHERE employee_id = p_emp_id;
EXCEPTION
  WHEN NO_DATA_FOUND THEN
    RAISE_APPLICATION_ERROR(-20003, 'Aucun employe trouve pour
        l''ID : ' || p_emp_id);
END query_emp;
/
```

b. Invoquer avec variables hôte (employé 120)

En SQL*Plus ou SQL Developer (bloc anonyme avec DBMS_OUTPUT) :

```
SET SERVEROUTPUT ON

DECLARE
  g_sal   employees.salary%TYPE;
g_job_id employees.job_id%TYPE;
BEGIN
  query_emp(120, g_sal, g_job_id);
  DBMS_OUTPUT.PUT_LINE('Salaire : ' || g_sal);
  DBMS_OUTPUT.PUT_LINE('Job ID : ' || g_job_id);
END;
```

```
/
```

Avec variables hôte SQL*Plus (équivalent demandé) :

```
VARIABLE g_sal NUMBER
VARIABLE g_job_id VARCHAR2(20)

EXEC query_emp(120, :g_sal, :g_job_id)

PRINT g_sal g_job_id
```

Résultat attendu pour l'employé 120 : par ex. salaire 8000, job_id ST_MAN.

c. Appel avec EMPLOYEE_ID = 300

```
EXEC query_emp(300, :g_sal, :g_job_id);
-- ou dans un bloc PL/SQL :
-- query_emp(300, g_sal, g_job_id);
```

Comportement : erreur (NO_DATA_FOUND).
Pourquoi ? Aucun enregistrement dans `EMPLOYEES` avec `employee_id = 300`. Le `SELECT INTO` ne renvoie aucune ligne, donc Oracle lève `NO_DATA_FOUND`, capturée puis relayée par `RAISE_APPLICATION_ERROR(-20003, ...)`.

Récapitulatif des procédures

Procédure	Rôle	Paramètres	Exception / remarque
ADD_JOB	Insérer un emploi	IN job_id, IN job_title	DUP_VAL_ON_INDEX si job_id existe déjà
UPD_JOB	Modifier le titre	IN job_id, IN job_title	Vérifier SQL%ROWCOUNT si 0 ligne mise à jour
DEL_JOB	Supprimer un emploi	IN job_id	Vérifier SQL%ROWCOUNT si 0 ligne supprimée

Procédure	Rôle	Paramètres	Exception / remarque
QUERY_EMP	Retourner salaire, job_id	IN emp_id, OUT salary, OUT job_id	NO_DATA_FOUND si employé inexistant

Résolution TP Pratique 3

1. Fonction Q_JOB

a. Création – retour du titre d'emploi

```
CREATE OR REPLACE FUNCTION q_job (
p_job_id IN jobs.job_id%TYPE
) RETURN
jobs.job_title%TYPE IS
  v_title jobs.job_title%TYPE;
BEGIN
  SELECT job_title
    INTO v_title
    FROM jobs
   WHERE job_id = p_job_id;
  RETURN v_title;
EXCEPTION
  WHEN NO_DATA_FOUND THEN
    RETURN
NULL;
END q_job;
/
```

b. Variable hôte et appel (SA_REP)

```
VARIABLE g_title VARCHAR2(35)
EXEC :g_title := q_job('SA_REP')

PRINT g_title
```

Résultat attendu :

G_TITLE
Sales Representative

2. Fonction ANNUAL_COMP

a. Création – salaire annuel avec gestion des NULL

Formule : **salaire_annuel = (salaire × 12) + (commission_pct × salaire × 12)**
Les NULL sont traités comme 0 pour que le résultat ne soit jamais NULL.

```
CREATE OR REPLACE FUNCTION annual_comp (
p_salary        IN employees.salary%TYPE,
p_commission_pct IN
employees.commission_pct%TYPE
) RETURN NUMBER
IS
BEGIN
  RETURN (NVL(p_salary, 0) * 12)
       + (NVL(p_commission_pct, 0) * NVL(p_salary, 0) * 12);
END annual_comp;
/
```

b. Utilisation dans un SELECT (département 80)

```
SELECT employee_id,
last_name,
       annual_comp(salary, commission_pct) AS "Annual
Compensation"
  FROM employees
 WHERE department_id = 80
 ORDER BY "Annual Compensation" DESC;
```

Exemple de résultat (valeurs possibles) :

EMPLOYEE_ID	LAST_NAME	Annual Compensation
145	Russell	235200
146	Partners	210600
147	Errazuriz	187200
148	Cambrault	175300
149	Zlotkey	151200
150	Livingston	120960
151	Johnson	81840

3. Fonction VALID_DEPTID et procédure NEW_EMP

a. Fonction VALID_DEPTID (retour BOOLEAN)

```
CREATE OR REPLACE FUNCTION valid_deptid (
p_dept_id IN
departments.department_id%TYPE
```

```
) RETURN
BOOLEAN IS
  v_count NUMBER;
BEGIN
  SELECT
COUNT(*)
    INTO v_count
    FROM departments
   WHERE department_id = p_dept_id;

  RETURN (v_count > 0);
END valid_deptid;
/
```

b. Procédure NEW_EMP avec valeurs par défaut

Valeurs par défaut : commission 0, salaire 1000, département 30, job_id SA_REP, manager_id 145. ID employé via **EMPLOYEES_SEQ.NEXTVAL**. Paramètres obligatoires : nom, prénom, e-mail.

```
CREATE OR REPLACE PROCEDURE new_emp (    p_last_name
IN  employees.last_name%TYPE,      p_first_name    IN
employees.first_name%TYPE,     p_email            IN
employees.email%TYPE,          p_commission_pct   IN
employees.commission_pct%TYPE DEFAULT 0,    p_salary
IN      employees.salary%TYPE      DEFAULT      1000,
p_department_id    IN    employees.department_id%TYPE
DEFAULT 30,   p_job_id      IN employees.job_id%TYPE
DEFAULT    'SA_REP',           p_manager_id       IN
employees.manager_id%TYPE DEFAULT 145
) IS
  v_emp_id employees.employee_id%TYPE;
BEGIN
  IF NOT valid_deptid(p_department_id) THEN
    RAISE_APPLICATION_ERROR(-20010,
      'Departement inexistant (ID : ' || p_department_id || ').
Employe non ajoute.');   END IF;   v_emp_id :=
employees_seq.NEXTVAL;

  INSERT INTO employees
(     employee_id,
first_name,
    last_name,
email,
hire_date,
```

```
job_id,
salary,
commission_pct,
manager_id,
department_id   )
VALUES (
v_emp_id,
p_first_name,
p_last_name,
p_email,
SYSDATE,
p_job_id,
p_salary,
p_commission_pct,
p_manager_id,
p_department_id
);

  COMMIT;
  DBMS_OUTPUT.PUT_LINE('Employe ajoute : ' || v_emp_id || ' - '
        || p_first_name || ' ' || p_last_name);
EXCEPTION
  WHEN OTHERS THEN
    ROLLBACK;
    RAISE;
END new_emp;
/
```

c. Test – Jane Harris, département 15

```
SET SERVEROUTPUT ON

EXEC new_emp('Harris', 'Jane', 'JHARRIS', p_department_id => 15);
```

Résultat attendu : erreur (ou message) car le département **15** n'existe pas dans **DEPARTMENTS** (VALID_DEPTID retourne FALSE). Message du type : *Departement inexistant (ID : 15). Employe non ajoute.* Aucune ligne insérée.

d. Test – Joe Harris, département 80

```
EXEC new_emp('Harris', 'Joe', 'JOHARRIS', p_department_id => 80);
```

Résultat attendu : succès. Le département **80** existe, VALID_DEPTID retourne TRUE, une ligne est insérée dans EMPLOYEES avec les valeurs par défaut (salaire 1000, job SA_REP, manager 145, etc.) et un nouvel employee_id fourni par EMPLOYEES_SEQ. Vérification :

```
SELECT employee_id, first_name, last_name, email, department_id,
       job_id, salary
```

```
  FROM employees  WHERE last_name =
'Harris' AND first_name = 'Joe';
```