2024

INSTITUT SUPÉRIEUR DE GESTION DE TUNIS
Since 1969

INSTITU SUPÉRIEUR DE GESTION

RAPPORT DU PROJET "SGBD"

Nour Charfeddine / Nader ben ammar

APPERÇUE DES TABLES

	NUMEXEMPLAIRE	NUMFILM	CODESUPPORT	VO	PROBLEME	DETAILSUPPORT
1	1001	101	DVD	Oui	(null)	Bonne condition
2	1002	101	VHS	Non	Rayures	Besoin de réparation
3	1003	102	DVD	Oui	(null)	Excellent état

	NUMFILM	TITRE	REALISATEUR
1	101	Jurassic Park	1
2	102	Inception	2
3	103	The Godfather	3

	NUMINDIVIDUE	NOMINDIVIDU	PRENOMINDIVIDU
1	1	Spielberg	Steven
2	2	Nolan	Christopher
3	3	Coppola	Francis

	LOGIN	NOMCLIENT	PRENOMCLIENT	MOTDEPASSE	ADRESSE
1	john_doe	Doe	John	secret123	123 Rue des Roses
2	jane_smith	Smith	Jane	password456	456 Avenue des Lilas

	NUMEXEMPLAIRE	DATELOCATION	LOGIN	DATEENVOI	DATERETOUR
1	1001	01/05/24	john_doe	01/05/24	05/05/24
2	1002	02/05/24	jane_smith	02/05/24	06/05/24

CODE D'IMPLÉMENTATION

```
Création des tables
create table individu(
numindividue number(8) primary key,
nomindividu varchar(50) not null,
prenomindividu varchar(50) not null);
CREATE TABLE film(
numfilm number(8) primary key,
titre varchar(50) not null,
realisateur number(8)references individu(numindividue));
create table exemplaire(
numexemplaire number(8) primary key,
numfilm number(8)references film(numfilm),
codesupport varchar(8) not null,
vo varchar(8),
probleme varchar(1000),
detailsupport varchar(1000));
create table clientt(
login varchar(30) primary key,
nomclient varchar(50) not null,
prenomclient varchar(50) not null,
motdepasse varchar(20),
adresse varchar(70));


create table LOCATION(
numexemplaire number(8) references exemplaire(numexemplaire),
datelocation date,
login varchar(30) references clientt(login),
dateenvoi date not null,
dateretour date not null,
primary key(numexemplaire,datelocation));
```

REPLISSAGE DE LA BASE

```sql
-- Insertions
INSERT INTO individu (numindividue, nomindividu, prenomindividu) VALUES (1, 'Spielberg', 'Steven');
INSERT INTO individu (numindividue, nomindividu, prenomindividu) VALUES (2, 'Nolan', 'Christopher');
INSERT INTO individu (numindividue, nomindividu, prenomindividu) VALUES (3, 'Coppola', 'Francis');

DESC exemplaire;

INSERT INTO film (numfilm, titre, realisateur) VALUES (101, 'Jurassic Park', 1); -- Réalisé par Spielberg
INSERT INTO film (numfilm, titre, realisateur) VALUES (102, 'Inception', 2); -- Réalisé par Nolan
INSERT INTO film (numfilm, titre, realisateur) VALUES (103, 'The Godfather', 3); -- Réalisé par Coppola

INSERT INTO exemplaire (numexemplaire, numfilm, codesupport, vo, probleme, detailsupport)
VALUES (1001, 101, 'DVD', 'Oui', NULL, 'Bonne condition');
INSERT INTO exemplaire (numexemplaire, numfilm, codesupport, vo, probleme, detailsupport)
VALUES (1002, 101, 'VHS', 'Non', 'Rayures', 'Besoin de réparation');
INSERT INTO exemplaire (numexemplaire, numfilm, codesupport, vo, probleme, detailsupport)
VALUES (1003, 102, 'DVD', 'Oui', NULL, 'Excellent état');


INSERT INTO clientt (login, nomclient, prenomclient, motdepasse, adresse) VALUES ('john_doe', 'Doe', 'John', 'secret123', '123 Rue
INSERT INTO clientt (login, nomclient, prenomclient, motdepasse, adresse) VALUES ('jane_smith', 'Smith', 'Jane', 'password456', '45
```

```sql
INSERT INTO LOCATION (numexemplaire, datelocation, login, dateenvoi, dateretour) VALUES (1001, TO_DATE('2024-05-01', 'YYYY-MM-DD'), 'john_
INSERT INTO LOCATION (numexemplaire, datelocation, login, dateenvoi, dateretour) VALUES (1002, TO_DATE('2024-05-02', 'YYYY-MM-DD'), 'jane_
```

1/ FONCTION NBREFILM()

Cette fonction prend comme paramètre l'id d'un réalisateur et retourne le nombre des filmes réalisés

FONCTION

```
CREATE OR REPLACE FUNCTION nbreFilms(realisateur_id number)
RETURN number
IS
    total_films number;
BEGIN
    SELECT COUNT(*)
    INTO total_films
    FROM film
    WHERE realisateur = realisateur_id;

    RETURN total_films;
END;
```

RÉSULTAT

2/ GESTION DE BONUS

A/ CRÉATION DE LA TABLE BONUS

```sql
--test Q2/a
CREATE TABLE tableBonus (
    login VARCHAR2(30) PRIMARY KEY,
    bonus NUMBER(8,2),
    nbrExLoues NUMBER(8)
);
```

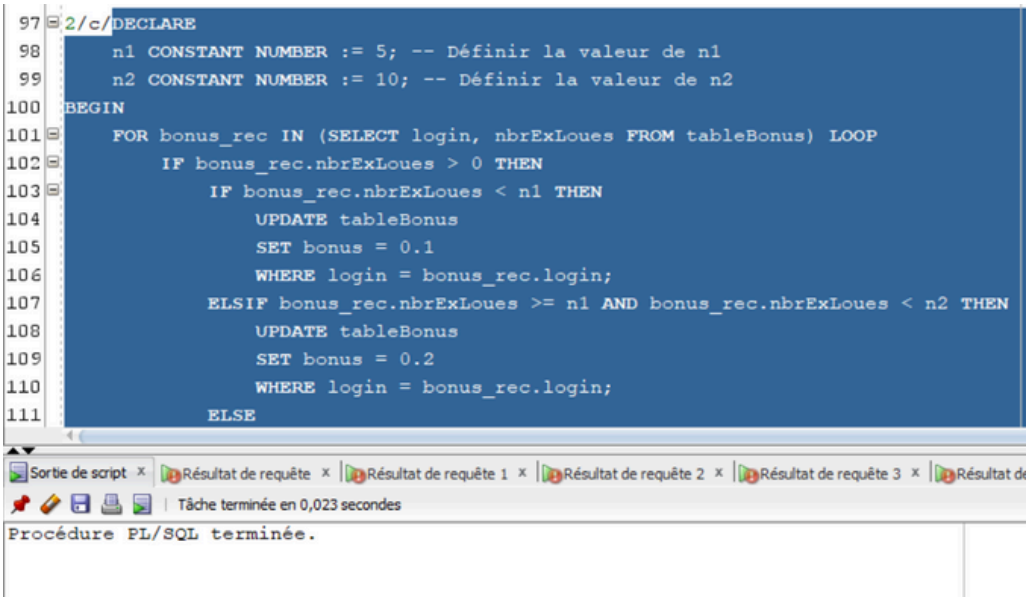
B/ RÉCUPÉRATION À PARTIR DE LA BASE DE DONNÉES LES VALEURS CORRESPONDANT AUX LOGINS

```sql
DECLARE
    result1 NUMBER;
    result2 NUMBER;
    result3 NUMBER;
BEGIN
    -- Appel de la fonction avec différents scénarios

    -- Scénario 1: n1 = 10, n2 = 5 (n2 non nul)
    result1 := Pourcentage(10, 5);
    DBMS_OUTPUT.PUT_LINE('Pourcentage(10, 5) = ' || result1);

    -- Scénario 2: n1 = 0, n2 = 8 (n1 nul)
    result2 := Pourcentage(0, 8);
    DBMS_OUTPUT.PUT_LINE('Pourcentage(0, 8) = ' || result2);

    -- Scénario 3: n1 = 12, n2 = 0 (n2 nul)
    result3 := Pourcentage(12, 0);
    DBMS_OUTPUT.PUT_LINE('Pourcentage(12, 0) = ' || result3);

EXCEPTION
    WHEN OTHERS THEN
        DBMS_OUTPUT.PUT_LINE('Une erreur s\'est produite lors de l\'exécution de la fonction Pourcentage.');
END;
```

RÉSULTAT DE LA TABLE BONUS

	LOGIN	BONUS	NBREXLOUES
1	john_doe	0,1	3
2	jane_smith	0,2	8
3	utilisateur3	0,4	12

C/
CE BLOC PL/SQL PARCOURT LA TABLE TABLEBONUS, METTANT À JOUR LES BONUS EN FONCTION DES CONDITIONS SPÉCIFIÉES.



```
2/c/DECLARE
    n1 CONSTANT NUMBER := 5; -- Définir la valeur de n1
    n2 CONSTANT NUMBER := 10; -- Définir la valeur de n2
BEGIN
    FOR bonus_rec IN (SELECT login, nbrExLoues FROM tableBonus) LOOP
        IF bonus_rec.nbrExLoues > 0 THEN
            IF bonus_rec.nbrExLoues < n1 THEN
                UPDATE tableBonus
                SET bonus = 0.1
                WHERE login = bonus_rec.login;
            ELSIF bonus_rec.nbrExLoues >= n1 AND bonus_rec.nbrExLoues < n2 THEN
                UPDATE tableBonus
                SET bonus = 0.2
                WHERE login = bonus_rec.login;
            ELSE
```

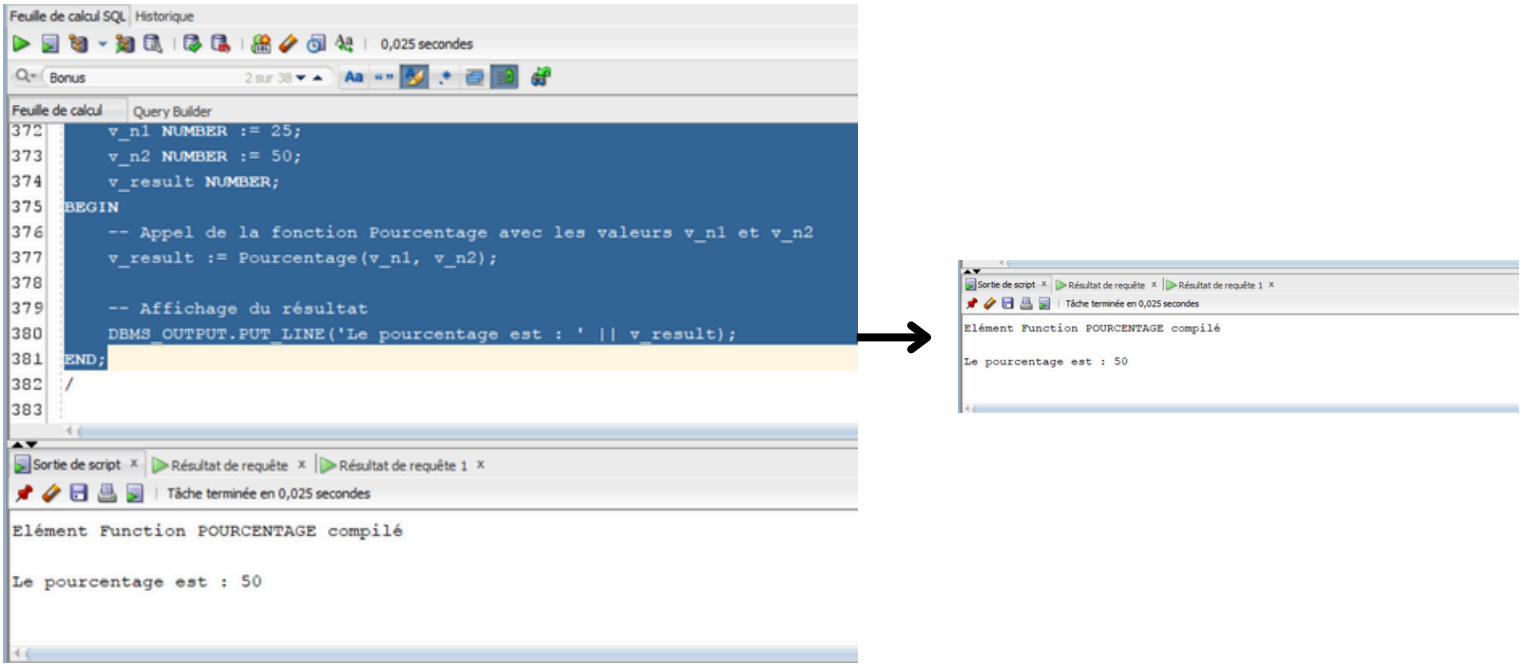
Sortie de script | Résultat de requête | Résultat de requête 1 | Résultat de requête 2 | Résultat de requête 3

Tâche terminée en 0,023 secondes

```
Procédure PL/SQL terminée.
```

3/ STATISTIQUES POUR LA VIDÉOTHÈQUE

A/
CETTE FONCTIONNOMMÉE “ "POURCENTAGE" CALCULE LE POURCENTAGE DE N1 PAR RAPPORT À N2.



Feuille de calcul SQL | Historique

0,025 secondes

Bonus

Feuille de calcul | Query Builder

```
    v_n1 NUMBER := 25;
    v_n2 NUMBER := 50;
    v_result NUMBER;
BEGIN
    -- Appel de la fonction Pourcentage avec les valeurs v_n1 et v_n2
    v_result := Pourcentage(v_n1, v_n2);

    -- Affichage du résultat
    DBMS_OUTPUT.PUT_LINE('Le pourcentage est : ' || v_result);
END;
/
```

Sortie de script | Résultat de requête | Résultat de requête 1

Tâche terminée en 0,025 secondes

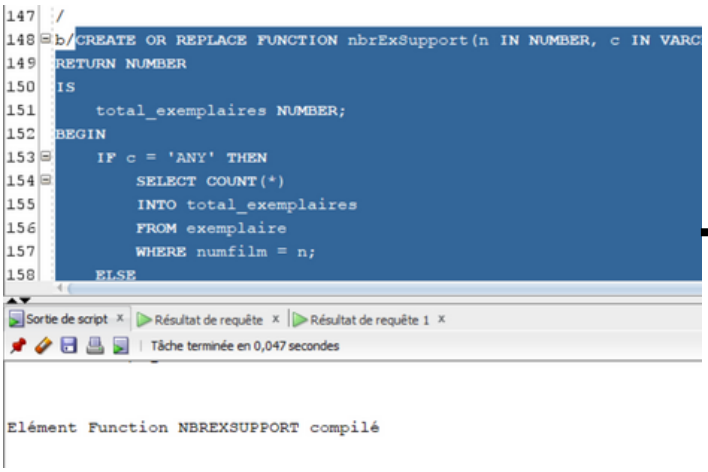
```
Elément Function POURCENTAGE compilé

Le pourcentage est : 50
```

Sortie de script | Résultat de requête | Résultat de requête 1

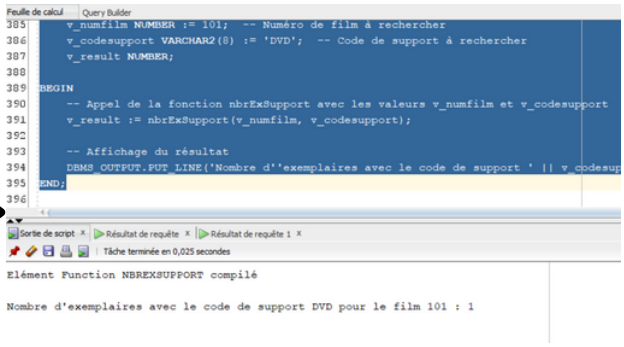
Tâche terminée en 0,025 secondes

```
Elément Function POURCENTAGE compilé

Le pourcentage est : 50
```

B/
CETTE FONCTION PREND EN PARAMÈTRE UN NUMÉRO DE FILM ET UN CODE DE SUPPORT, ET RETOURNE LE NOMBRE D'EXEMPLAIRES DU FILM AYANT CE SUPPORT. SI LE PARAMÈTRE DU SUPPORT EST "ANY", UNE EXCEPTION EST LEVÉE CAR CELA N'EST PAS AUTORISÉ. EN CAS DE NON-TROUVAILLE D'EXEMPLAIRES, LA FONCTION RETOURNE 0.

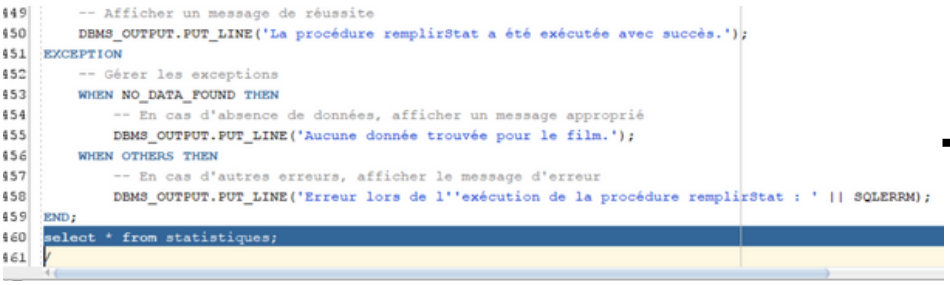


COMPILATION

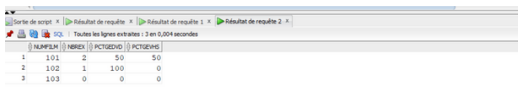


EXÉCUTION

D/
CETTE PROCÉDURE REMPLIT LA TABLE STATISTIQUES AVEC LE NOMBRE D'EXEMPLAIRES DE CHAQUE FILM, AINSI QUE LE POURCENTAGE D'EXEMPLAIRES DE TYPE DVD ET DE TYPE VHS POUR CHAQUE FILM.



COMPILATION



EXÉCUTION

4A/ LE NOMBRE DE FILMS POUR LESQUELS ON NE POSSÈDE AUCUN EXEMPLAIRE

COMPILATION

```
233  *******
234  4/a/CREATE OR REPLACE PACKAGE statistics AS
235      PROCEDURE count_films_no_copies(p_count OUT NUMBER);
236  END statistics;
237  /
238
239  CREATE OR REPLACE PACKAGE BODY statistics AS
240      PROCEDURE count_films_no_copies(p_count OUT NUMBER) IS
```

Sortie de script | Résultat de requête | Résultat de requête 1 | Résultat de requête 2

Tâche terminée en 0,057 secondes

```
Elément Package STATISTICS compilé


Elément Package STATISTICS compilé


Elément Package Body STATISTICS compilé
```

EXÉCUTION

```
233  *******
234  4/a/
235  CREATE OR REPLACE PACKAGE BODY statistics AS
236      PROCEDURE count_films_no_copies(p_count OUT NUMBER) IS
237      BEGIN
238          SELECT COUNT(*) INTO p_count
239          FROM film f
240          WHERE NOT EXISTS (
```

Sortie de script | Résultat de requête | Résultat de requête 1 | Résultat de requête 2

Tâche terminée en 0,027 secondes

```
Elément Package Body STATISTICS compilé

Number of films without copies: 1


Procédure PL/SQL terminée.
```

4B/ LE POURCENTAGE D'EXEMPLAIRES POUR LESQUELS ON N'A JAMAIS SIGNALÉ DE PROBLÈME.

COMPILATION

```
--test Q4 b
DECLARE
    pourcentage NUMBER;
BEGIN
    pourcentage := statistics.pct_exemplaires_sans_probleme;

    IF pourcentage IS NOT NULL THEN
```

EXÉCUTION

```
Elément Package Body STATISTICS compilé

Pourcentage d'exemplaires sans problème : 66,66666666666666666666666666666666666667%


Procédure PL/SQL terminée.
```

5/ CETTE FONCTION COMPTE LE NOMBRE DE LIGNES DANS LA TABLE TRACE. ELLE UTILISE UNE DÉCLARATION DE VARIABLE, UNE REQUÊTE SELECT POUR COMPTER LES LIGNES ET STOCKER LE RÉSULTAT DANS CETTE VARIABLE, PUIS RETOURNE LE NOMBRE DE LIGNES COMPTÉES. EN CAS D'ERREUR, ELLE RETOURNE 0.

```
Valeurs ajoutées dans la table TRACE
Lorsqu'une insertion est effectuée dans la table LOCATION, voici ce qui se passe en fonction des déclencheurs :

Déclencheur TRIG_1 (BEFORE INSERT) : Pour chaque ligne insérée, une nouvelle ligne est ajoutée à la table TRACE avec le message 'Trigger 1'.
Déclencheur TRIG_2 (BEFORE INSERT) : Ce déclencheur tenterait également d'ajouter une ligne à la table TRACE avec le message 'Trigger 2',
mais il échouera en raison de l'erreur liée à nbValLog non définie.
Déclencheur TRIG_3 (AFTER INSERT) : Après chaque insertion, une nouvelle ligne est ajoutée à la table TRACE avec le message 'Trigger 3'.
Déclencheur TRIG_4 (AFTER INSERT) : Après chaque insertion, une nouvelle ligne est ajoutée à la table TRACE avec le message 'Trigger 4'.

Ordre d'activation des déclencheurs
L'ordre d'activation des déclencheurs dépend de leur type (BEFORE ou AFTER) et de leur numéro d'ordre d'activation :

Déclencheurs BEFORE INSERT : Ils sont activés avant l'insertion effective dans la table. Dans ce cas, TRIG_1 s'exécutera avant TRIG_2 pour
chaque ligne insérée.
Déclencheurs AFTER INSERT : Ils sont activés après l'insertion effective dans la table. TRIG_3 s'exécutera avant TRIG_4 pour chaque ligne
insérée.
```

```
END;

select * from statistiques;
--test Q5 partie 1
DECLARE
    v_nbVal INTEGER;
BEGIN
    v_nbVal := nbValLog();
    DBMS_OUTPUT.PUT_LINE('Nombre de lignes dans la table trace : ' || v_nbV
END;

SELECT nbValLog() AS nombre_de_lignes FROM DUAL;
```

Sortie de script | Résultat de requête

Toutes les lignes extraites : 1 en 0,006 secondes

	NOMBRE_DE_LIGNES
1	0

→ Table TRACE créé(e).

MERCI

Nour Charfeddine-Nader Ben Ammar