Mini projet Oracle PL/SQL

Réalisé par :

Chhaidar Ayoub

&

Ferchichi Rami

Tout d'abord on a créé un utilisateur C##AB et une base nommée sgbd ,après on a implémenter les tableaux suivants :

|  |  |
| --- | --- |
| Les tables | Pk et FGN key |
| CREATE TABLE "C##AB"."CLIENT"  ( "LOGIN" VARCHAR2(35 BYTE),  "NOMCLIENT" VARCHAR2(30 BYTE),  "PRENOMCLIENT" VARCHAR2(30 BYTE),  "MOTDEPASSE" VARCHAR2(20 BYTE),  "ADRESSE" VARCHAR2(50 BYTE)  ); | * CREATE UNIQUE INDEX "C##AB"."PK\_CLIENT" ON "C##AB"."CLIENT" ("LOGIN"); |
| CREATE TABLE "C##AB"."EXEMPLAIRE"  ( "NUMEXEMPLAIRE" NUMBER,  "NUMFILM" NUMBER,  "CODESUPPORT" VARCHAR2(30 BYTE),  "VO" VARCHAR2(20 BYTE),  "PROBLEME" VARCHAR2(50 BYTE),  "DETAILSUPPORT" VARCHAR2(30 BYTE)  ); | * CREATE UNIQUE INDEX "C##AB"."EXP\_PK" ON "C##AB"."EXEMPLAIRE" ("NUMEXEMPLAIRE"); * ALTER TABLE "C##AB"."EXEMPLAIRE" ADD CONSTRAINT "EXP\_FK" FOREIGN KEY ("NUMFILM"); |
| CREATE TABLE "C##AB"."FILM"  ( "NUMFILM" NUMBER,  "TITRE" VARCHAR2(30 BYTE),  "REALISATEUR" NUMBER  ); | * CREATE UNIQUE INDEX "C##AB"."FILM\_PK" ON "C##AB"."FILM" ("NUMFILM"); * ALTER TABLE "C##AB"."FILM" ADD CONSTRAINT "FILM\_FK" FOREIGN KEY ("REALISATEUR"); |
| CREATE TABLE "C##AB"."INDIVIDU"  ( "NUMINDIVIDU" NUMBER,  "NOMINDIVIDU" VARCHAR2(30 BYTE),  "PRNOMINDIVIDU" VARCHAR2(20 BYTE)  ); | * CREATE UNIQUE INDEX "C##AB"."IND\_PK" ON "C##AB"."INDIVIDU" ("NUMINDIVIDU"); |
| CREATE TABLE "C##AB"."LOCATION"  ( "NUMEXEMPLAIRE" NUMBER,  "DATELOCATION" DATE,  "LOGIN" VARCHAR2(35 BYTE),  "DATEENVOI" DATE,  "DATERETOUR" DATE  ); | * CREATE UNIQUE INDEX "C##AB"."LOC\_PK" ON "C##AB"."LOCATION" ("DATELOCATION"); * ALTER TABLE "C##AB"."LOCATION" ADD CONSTRAINT "LOC\_FK" FOREIGN KEY ("NUMEXEMPLAIRE"); * ALTER TABLE "C##AB"."LOCATION" ADD CONSTRAINT "LOC\_FK" FOREIGN KEY ("NUMEXEMPLAIRE"); |

Après on a peupler cette base de données.

**1. on a crée la fonction NBREFILMS :**

CREATE OR REPLACE FUNCTION "C##AB"."NBREFILMS" (numind NUMBER)

RETURN NUMBER

IS

result NUMBER := 0;

BEGIN

SELECT COUNT(realisateur) INTO result

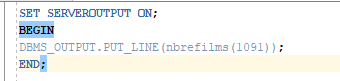
FROM FILM

WHERE realisateur = numind;

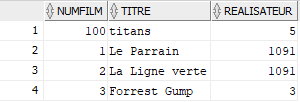
RETURN result;

END;

* après on a tester la fonction avec le réalisateur numéro 1091



* Le résultat :



**2. a. on créer la table tableBonus**

CREATE TABLE "C##AB"."TABLEBONUS"

( "LOGIN" VARCHAR2(35 BYTE),

"BONUS" NUMBER(8,2),

"NBREXLOUES" NUMBER(8,0)

);

**b. remplisage de la table tableBonus**

INSERT INTO tableBonus (login, nbrExLoues) SELECT c.login, COUNT(\*) FROM client c INNER JOIN location l ON c.login = l.login WHERE l.dateLocation BETWEEN TO\_DATE('01/01/2022', 'DD/MM/YYYY') AND TO\_DATE('31/12/2022', 'DD/MM/YYYY') GROUP BY c.login;

**c.mise a jour du tableBonus**

create or replace PROCEDURE updatebonus(n1 number ,n2 number )

IS

BEGIN

UPDATE tablebonus SET bonus = CASE WHEN 0< nbrExLoues and nbrExLoues < n1 THEN 0.1 WHEN n1<=nbrExLoues and nbrExLoues < n2 THEN 0.2 WHEN nbrExLoues <= n2 THEN 0.4 ELSE 0

END;

END updatebonus;   
  
**d. Affichage de nom, son prénom et son bonus Pour chaque client.**

select client.nomclient,client.prenomclient ,bonus from client JOIN tablebonus ON client.login=tablebonus.login ;

**3.a. fonction Pourcentage :**

CREATE OR REPLACE FUNCTION Pourcentage(n1 INTEGER, n2 INTEGER)

RETURN NUMBER IS result NUMBER;

BEGIN

IF n2 = 0 THEN RAISE\_APPLICATION\_ERROR(-20000, 'Division by zero is not allowed.');

END IF;

result := ROUND((n1 / n2) \* 100, 2);

RETURN result;

END;

**b. Fonction nbrExSupport:**

create or replace FUNCTION "NBREXSUPPORT" (n film.numfilm%TYPE, c exemplaire.codesupport%TYPE)

RETURN NUMBER IS

result NUMBER:=0;

BEGIN

IF (c = 'ANY') THEN

SELECT COUNT(\*) INTO result FROM exemplaire WHERE numfilm = n;

ELSE

SELECT COUNT(\*) INTO result FROM exemplaire WHERE numfilm = n AND codesupport = c;

IF (result = 0) THEN

RAISE\_APPLICATION\_ERROR(-20001, 'Le code de support n'existe pas ou film sans exemplaire');

END IF;

END IF;

RETURN result;

END;

**c. table statistiques :**

CREATE TABLE "C##AB"."STATISTIQUES"

( "NUMFILM" NUMBER,

"NBREEX" NUMBER(\*,0),

"PCTGEDVD" FLOAT(126),

"PCTGEVHS" FLOAT(126)

);

**d. une procédure remplirStat:**

create or replace PROCEDURE "REMPLIRSTAT" IS

CURSOR c1 IS SELECT distinct numfilm FROM exemplaire;

dvd STATISTIQUES.PCTGEDVD%TYPE;

anyy VARCHAR2(20) := 'ANY';

alll STATISTIQUES.NBREEX%TYPE;

vhs STATISTIQUES.PCTGEVHS%TYPE;

BEGIN

FOR row IN c1 LOOP

alll := nbrExSupport(ROW.numfilm,'ANY');

dvd:= Pourcentage(nbrExSupport(ROW.numfilm,'DVD'),nbrExSupport(ROW.numfilm,'ANY'));

vhs :=Pourcentage(nbrExSupport(ROW.numfilm,'VHS'),nbrExSupport(ROW.numfilm,'ANY'));

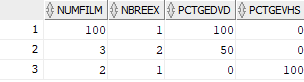
INSERT INTO statistiques (numfilm, nbreex, pctgedvd, pctgevhs)

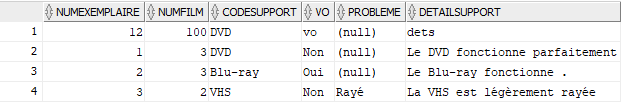
VALUES (row.numfilm, alll,dvd ,vhs );

END LOOP;

END;

* Résultat après exécuter remplirstat() et table exemplaire associe au fonction:





**4 .un package statistics :**

* Head:

create or replace PACKAGE "STATISTICS" AS

FUNCTION nbreFillmNoExp RETURN NUMBER;

FUNCTION pourNonProb RETURN NUMBER;

END statistics ;

* Body:

create or replace PACKAGE BODY "STATISTICS" AS

FUNCTION nbreFillmNoExp

RETURN NUMBER

IS

result NUMBER;

BEGIN

result := 0;

SELECT COUNT(\*) INTO result FROM film WHERE numfilm NOT IN (SELECT DISTINCT numfilm FROM exemplaire);

RETURN result;

END nbreFillmNoExp;

FUNCTION pourNonProb

RETURN NUMBER

IS

nbreEx NUMBER;

result NUMBER;

nbreExnop NUMBER;

BEGIN

SELECT COUNT(\*) INTO nbreEx FROM exemplaire;

SELECT COUNT(\*) INTO nbreExnop FROM exemplaire WHERE probleme IS NULL OR probleme = '';

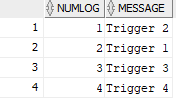
result := (nbreExnop/ nbreEx)\*100;

RETURN result;

END pourNonProb;

END statistics;

**5.Les valeurs ajoutées Lors d’une insertion dans la table location :**



* Les déclencheurs BEFORE sont activés avant les déclencheurs AFTER .
* le déclencheur BEFORE avec FOR EACH ROW est active après l’autre déclencheur,mais le déclencheur AFTER avec FOR EACH ROW est active avant l’autre déclencheur.

**6. le déclencheur qui empêche la suppression d’une ligne de la table location :**

**create or replace TRIGGER prevent\_delete\_location**

**BEFORE DELETE ON location**

**FOR EACH ROW**

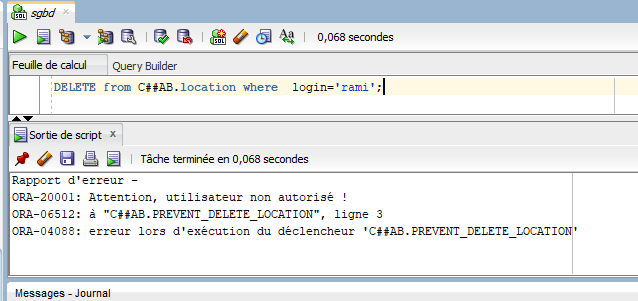
**BEGIN**

**IF user() != 'C##AB' THEN**

**RAISE\_APPLICATION\_ERROR(-20001, 'Attention, utilisateur non autorisé !');**

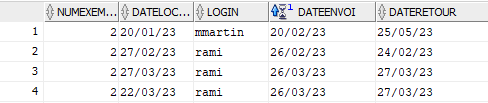
**END IF;**

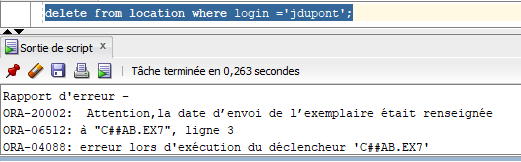
**END;**



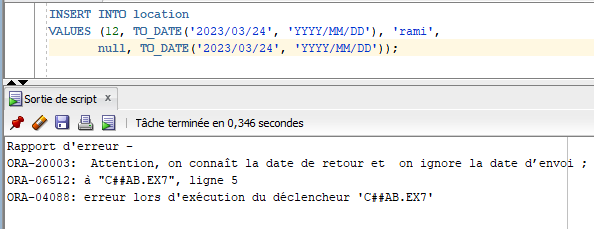
**7.**

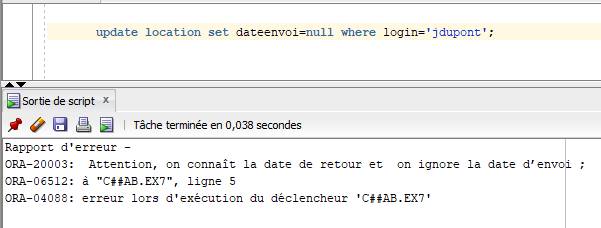
* **Refuse la suppression si la date d’envoi de l’exemplaire était renseignée :**



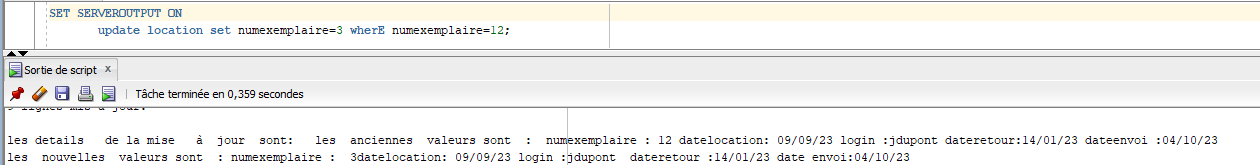


* **Refuse la mise à jour (ou insertion) qui aboutit à une location dont on connaît la date de retour et dont on ignore la date d’envoi :**





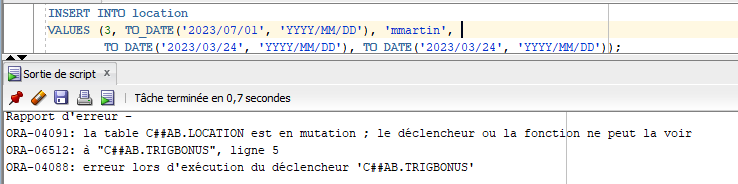
* Dans tous les autres cas, fait afficher un message précisant ce qui a été fait et pour quel login, numéro d’exemplaire et date de location :



**8.**

**a-**

lors d’une insertion dans la table location :



lors de l’insertion des données dans une table alors qu'un déclencheur ou une fonction est en cours d'exécution qui utilise cette même table.

Ce phénomène est appelé "table en mutation"

**b-**

