Objectif : Le but de ce TP est d'initier à la programmation PL/SQL de MySQL : Procédures et Fonctions Stockées, Déclencheurs (Trigger),

Partie I: Création des Triggers, Procédures et Fonctions sous MySQL

La journalisation

Un journal est une séquence d'enregistrements décrivant les mises à jours effectuées par les transactions

C'est l'historique d'une exécution sur fichier séquentiel

Un journal est dit « physique » s'il garde la trace des modifications au niveau octet à l'intérieur des pages

Ex: <Ti, numPg, Depl, Long, img_avant, img_après>

Un journal est dit « logique » s'il garde la trace de la description de haut niveau des opérations de mise à jour

Ex: « insérer le tuple x dans la table T et mettre à jour les index »

Etape 03 : Triggers pour enregistrer l' historique des modifications d'une table.

Certaines bases de données proposent des fonctions PITR (Point in time recovery) pour retrouver l'état de la base de donnée à un instant précis, ou bien des systèmes de versions qui permettent de tracer l'historique d'une ligne dans une table. Ici, nous verrons un exemple simple d'utilisation de triggers pour enregistrer un historique des modifications d'une table.

a) Création :

```
CREATE TRIGGER Hist AFTER UPDATE ON Compte
FOR EACH ROW
BEGIN
  INSERT INTO JournalCompte(operation, date, Utilisateur,……)
  VALUES('update', NEW.NumCompte, NOW(), USER, OLD.sole,  NEW.solde);
END;
```

b) **GESTION DE TRIGGERS :**

DROP TRIGGER nomtrigger;
ALTER TABLE nomtable **DISABLE ALL TRIGGERS**;
ALTER TRIGGER nomtrigger **ENABLE**;
ALTER TABLE nomtable **ENABLE ALL TRIGGERS;**
ALTER TRIGGER nomtrigger **DISABLE**;
Show Triggers;
SHOW TRIGGERS FROM BD**;**
SELECT * FROM
INFORMATION_SCHEMA.TRIGGERS WHERE TRIGGER_SCHEMA='Nom BD'**;**

Etape 01 : Procédure d'inscription avec une date limite

a) Création

```
DROP PROCEDURE IF EXISTS Inscription;
CREATE PROCEDURE Inscription (parameter1 INT)
BEGIN
   DECLARE variable1 INT;
   SELECT COUNT(*) INTO variable1 FROM candidat;
   IF  DATEDIFF(SYSDATE(),'2008-11-30')<0 THEN
         INSERT into candidat (candidat. candidat id) values (parameter1);
   ELSE
      CALL  RAISE_APPLICATION_ERROR(-20002,'La capacité d''inscription à été atteinte !');
   END IF;
END;
```

b) Appels de procédures

call Inscription (30);

Etape 02 : Fonction : Calcul de factoriel d'un nombre entier

a) Création

```
CREATE FUNCTION factorial (n DECIMAL(3,0))
RETURNS DECIMAL(20,0)
DETERMINISTIC
BEGIN
      DECLARE factorial DECIMAL(20,0) DEFAULT 1;
      DECLARE counter DECIMAL(3,0);
      SET counter = n;
      factorial_loop: REPEAT
            SET factorial = factorial * counter;
            SET counter = counter - 1;
      UNTIL counter = 1
      END REPEAT;
      RETURN factorial;
END
```

b) **Utilisation des fonctions :**

- INSERT INTO NomTable VALUES (factorial(pi)) //
- SELECT s1, factorial (s1) FROM NomTable //
- UPDATE NomTable SET s1 = factorial(s1)
  WHERE factorial(s1) < 5 //

Etape 03 : Triggers pour enregistrer l' historique des modifications d'une table.

Certaines bases de données proposent des fonctions PITR (Point in time recovery) pour retrouver l'état de la base de donnée à un instant précis, ou bien des systèmes de versions qui permettent de tracer l'historique d'une

ligne dans une table. Ici, nous verrons un exemple simple d'utilisation de triggers pour enregistrer un historique des modifications d'une table.

c) Création :

```
CREATE TRIGGER Hist AFTER UPDATE ON Compte
FOR EACH ROW
BEGIN
  INSERT INTO JournalCompte(operation, date, Utilisateur,……)
  VALUES('update', NEW.NumCompte, NOW(), USER, OLD.sole,  NEW.solde);
END;
```

d) **GESTION DE TRIGGERS :**

DROP TRIGGER nomtrigger;
ALTER TABLE nomtable **DISABLE ALL TRIGGERS**;
ALTER TRIGGER nomtrigger **ENABLE**;
ALTER TABLE nomtable **ENABLE ALL TRIGGERS;**
ALTER TRIGGER nomtrigger **DISABLE**;
Show Triggers;
SHOW TRIGGERS FROM BD**;**
SELECT * FROM
INFORMATION_SCHEMA.TRIGGERS WHERE TRIGGER_SCHEMA='Nom BD'**;**

Partie II : Invocation depuis JAVA

Sous Eclipse créez un projet Java, ajouter la librairie mysql-connector-java-X.Y.Z-bin.jar et configurer le (à l'aide de l'assistant c'est plus rapide)

Et pour tester la procédure :

```
public class
SimpleTestProcedure {
public static void
main(String[] args)
throws
ClassNotFoundException, SQLException{Class.  forName ("com.mysql.jdbc.Driver");Connection con
 =DriverManager.
getConnection  ("jdbc:mysql://localhost:3306/test","root","");  ResultSet rs;
CallableStatement smt = con.prepareCall( "call simpleproc(?)"  ); smt.registerOutParameter(1,
java.sql.Types.INTEGER  );
rs = smt.executeQuery();
int  x = smt.getInt(1);smt.execute();System. out.println("Nombres de lignes affecé : "+x);}}
```