PL SQL

Séance 6

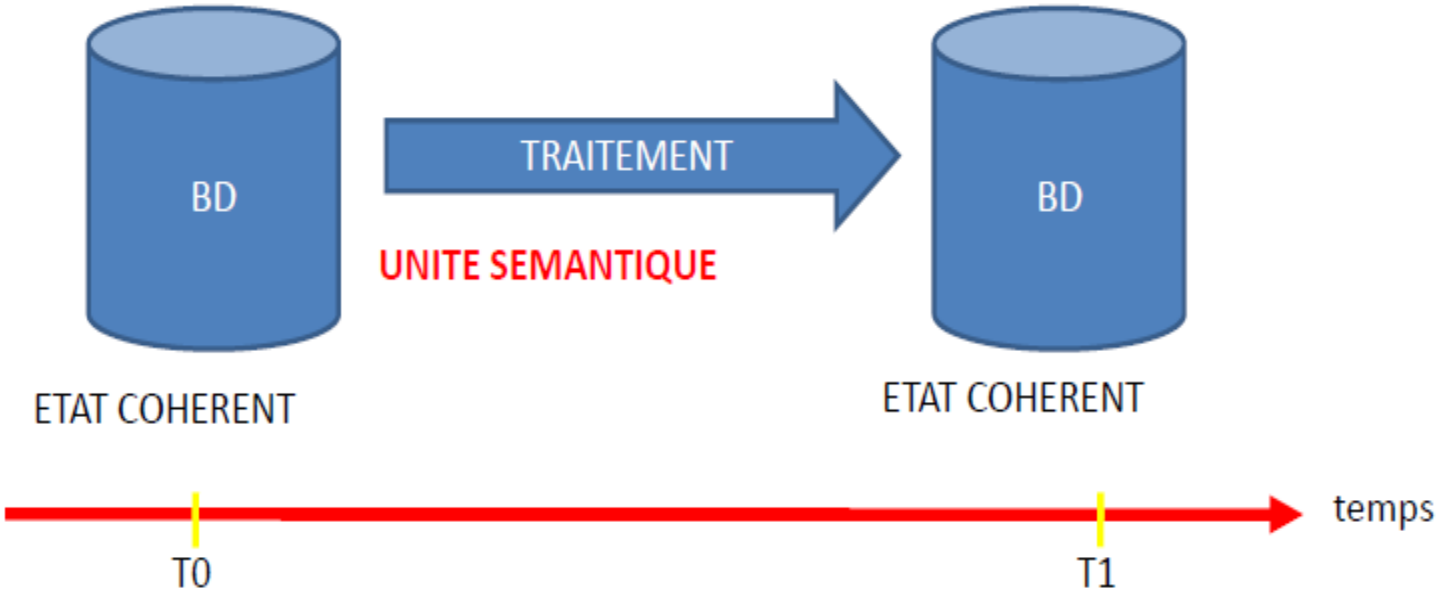
Pr. M’barek ELHALOUI

Révision/ Questions

- Les procédures (déclaration, paramètres IN, OUT, IN OUT)
- Les fonctions (paramètres IN, Retour de résultat)
- Les packages (Spécification, Corps)
- Questions ?

Les Transactions

Problème :



Les Transactions

Définition :

- Une transaction est un ensemble d'opérations associées (atomiques) , c'est-à-dire indivisibles.
- Nous considèrerons qu'un ensemble d'opérations est indivisible si une exécution partielle de ces instructions poserait des problèmes d'intégrité dans la base de données.

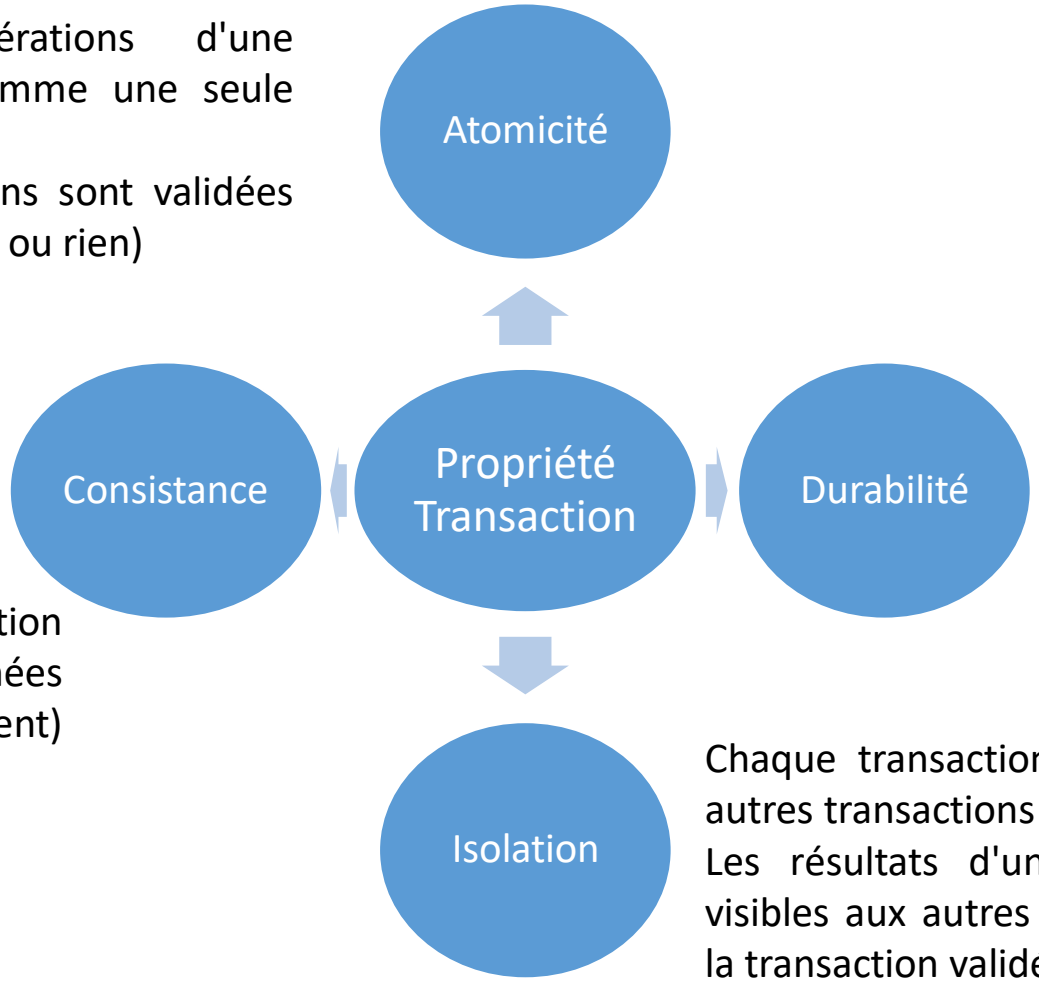
Exemple : Un virement d'un compte à un autre se fait en deux temps :

- Créditer un compte d'une somme S ,
- et débiter un autre de la même somme S

```
Update Comptes
    Set Solde:= Solde - 1000 where id_Customer = 5  ;
    Set Solde:= Solde + 1000 where id_Customer = 10 ;
Commit ;
```

Les Transactions

L'ensemble des opérations d'une transaction apparaît comme une seule opération atomique
Soit toutes les opérations sont validées ou toutes annulées (tout ou rien)



C'est la persistance des mises à jour d'une transaction validée.
Les effets d'une transaction validée sont durables et permanents, quelques soient les problèmes logiciels ou matériels, notamment après la fin de la transaction.

L'exécution de la transaction fait passer la base de données d'un état consistant (cohérent) à un autre état consistant

Chaque transaction est indépendante des autres transactions concurrentes.
Les résultats d'une transaction ne sont visibles aux autres transactions qu'une fois la transaction validée.

Les Transactions

Squelette d'une transaction :

```
/* instructions */
IF /* erreur */ THEN
        ROLLBACK;
ELSE
        COMMIT;
END;
```

- **COMMIT:** Cette instruction permet d'enregistrer **définitivement** dans la BD toutes les modifications effectuées au cours de la transaction.
- **Le ROLLBACK** annule toutes les modifications faites depuis le début de la transaction (donc depuis le précédent COMMIT).

Les Transactions

Exemple : Transaction avec traitement des exceptions

```
-- Debut de la transaction
Begin

    INSERT INTO EMPLOYE (ID,L_Name,F_Name,D_Naissance,SALARY)
    VALUES (1, 'Ahmed', 'Alami', '15/06/1990', 3500.00 );

    INSERT INTO EMPLOYE (ID,L_Name,F_Name,D_Naissance,SALARY)
    VALUES (2, 'Amine', 'Khalil', '23/08/1992', 3000.00 );

COMMIT;

    EXCEPTION
        WHEN DUP_VAL_ON_INDEX THEN
        raise_application_error (-20001,'duplicate ID');
        WHEN others THEN
        ROLLBACK ;
END;
-- Fin de la transaction
```

Les Transactions

- La variable d'environnement **AUTOCOMMIT** , qui peut être positionnée à ON ou à OFF permet d'activer la gestion des transactions.
- Si elle est positionnée à ON , chaque instruction a des répercussions immédiates sur la BD, sinon, **les modifications ne sont effectives qu'une fois qu'un COMMIT a été exécuté.**

Les Triggers

- **Un trigger (déclencheur)** est une procédure stockée qui se lance automatiquement lorsqu'un événement se produit.
- **Un événement** est toute modification des données se trouvant dans les tables.
- L'événement est en générale une instruction de type INSERT, UPDATE ou DELETE.
- On s'en sert pour **contrôler ou appliquer des contraintes** qu'il est impossible de formuler de façon déclarative.

Les Triggers

- **Un trigger peut être utilisé pour :**
  - l'intégrité des données
  - l'intégrité référentielle
  - la réplication de tables
  - le calcul automatique des données dérivées
  - L'historisation des événements
  - la sécurité
  - l'audit

Les Triggers

Type d'événement

- Lors de la création d'un trigger, il convient de préciser quel est le type d'événement qui le déclenche.
- Les principaux événements :
  - **INSERT**
  - **DELETE**
  - **UPDATE**

Moment de l'exécution

- On précise aussi si le trigger doit être exécuté avant (**BEFORE**) ou après (**AFTER**) l'événement.

Les Triggers

Types de triggers :

- **Triggers de table :** déclenchés une seule fois sur la table suite à un évènement.
- **Triggers de ligne (ROW) :** déclenchés pour chaque ligne de la table affectée par le trigger.
- Le trigger et de type ligne si l'option FOR EACH ROW est spécifiée, sinon c'est un trigger de table.

Les Triggers

- **Syntaxe :**

CREATE [OR REPLACE] TRIGGER *nom_Trigger* [BEFORE|AFTER] [INSERT

| UPDATE | DELETE] ON *nom_Table* [FOR EACH ROW|]

DECLARE *<déclarations>*

BEGIN

<Instructions>

END;

Les Triggers

Exemple 1 : Trigger sur Table

```
SQL> CREATE OR REPLACE TRIGGER ACCES_EMP
  2  BEFORE INSERT ON E_EMPLOYE
  3  BEGIN
  4   IF (TO_CHAR (sysdate,'DY') IN ('SAT','SUN'))
  5     OR (TO_CHAR(sysdate,'HH24')NOT BETWEEN
  6    '08' AND '18'
  7     THEN RAISE_APPLICATION_ERROR (-20500,
  8    'Vous ne pouvez pas utiliser la table E_EMPLOYE
        que pendant les heures normales.');
 10   END IF;
 11  END;
 12  /
```

Les Triggers

Traitement des exceptions :

- L'instruction **RAISE_APPLICATION_ERROR (code, message)** lève une exception sans nom portant un code et un message d'erreur message .
- Le trigger émet le message d'erreur lorsque la règle contrôlée n'est pas respectée.
- L'exécution de RAISE_APPLICATION_ERROR annule la transaction en cours.

Séparation des évènements

- Il est possible, en séparant les types d'événement par le mot-clé **OR**, de définir un trigger déclenché par plusieurs événements (Insert OR Update OR …).
- Les variables booléennes **INSERTING, UPDATING et DELETING** permettent d'identifier l'événement qui a déclenché le trigger.

Les Triggers

Exemple 2 : Trigger sur table

```
CREATE OR REPLACE TRIGGER ACCES_EMP
BEFORE INSERT OR UPDATE OR DELETE ON E_EMPLOYES
BEGIN
 IF (TO_CHAR (SYSDATE,'DY') IN ('SAT','SUN')) OR
    (TO_CHAR (SYSDATE, 'HH24') NOT BETWEEN '08' AND '18')
 THEN
   IF    DELETING THEN
     RAISE_APPLICATION_ERROR (-20502,'Vous ne pouvez pas    supprimer
dans la table E_EMPLOYE que pendant les heures       normales.');
   ELSIF  INSERTING THEN
     RAISE_APPLICATION_ERROR (-20500,'Vous ne pouvez pas ajouter    dans
la table E_EMPLOYE que pendant les heures    normales.');
   ELSIF   UPDATING ('SALAIRE') THEN
     RAISE_APPLICATION_ERROR (-20503,'Vous ne pouvez pas modifier   le
SALAIRE dans la table E_EMPLOYE que pendant les      heures normales.');
   ELSE
     RAISE_APPLICATION_ERROR (-20504, 'Vous ne pouvez pas
        modifier la table E_EMPLOYE que pendant les
        heures normales. ');
   END IF;
  END IF;
END;
```

Les Triggers

Accès aux lignes en cours de modification:

- Dans les triggers lignes (FOR EACH ROW) , il est possible avant la modification de chaque ligne, de lire l'ancienne ligne et la nouvelle ligne par l'intermédiaire des deux variables structurées **:old**.*colone* et **:new**.*colone*
- Par exemple le trigger suivant empêche de diminuer un salaire :

```
CREATE OR REPLACE TRIGGER pasDeBaisseDeSalaire
BEFORE UPDATE ON EMP
FOR EACH ROW
BEGIN
        IF (:old.sal > :new.sal) THEN
                RAISE_APPLICATION_ERROR(-20567,
                        'Pas de baisse de salaire !');
        END IF;
END;
```

Les Triggers

Exemple 4 : Trigger sur ligne

```
SQL>
SQL> CREATE OR REPLACE TRIGGER TRACE_EMPS
 2  AFTER DELETE ON E_EMPLOYE
 3  FOR EACH ROW
 4  BEGIN
 5              IF DELETING THEN
 6                      INSERT INTO TRACE_EMPS VALUES(
 7                              SEQ_TR_EMP.NEXTVAL,
 8                              SYSDATE,
 9                              USER,
10                              :OLD.NO,
11                              :OLD.NOM,
12                              :OLD.PRENOM,
13                              :OLD.DT_ENTREE,
14                              :OLD.TITRE,
15                              :OLD.SERVICE_NO,
16                              :OLD.COMMENTAIRE,
17                              :OLD.SALAIRE,
18                              :OLD.PCT_COMMISSION
19                      );
20              END IF;
21  END;
22  /
```

Traçabilité des suppressions sur une table

Les Triggers

Suppression :

- **DROP TRIGGER *nom_Trigger* ;**

Activation :

- **ALTER TRIGGER *nom_Trigger* ENABLE ;**
- **ALTER TABLE *nom_table* ENABLE ALL TRIGGERS ;**

Désactivation

- **ALTER TRIGGER *nom_Trigger* DISABLE ;**
- **ALTER TABLE *nom_table* DISABLE ALL TRIGGERS ;**

Exercices

- Créer un trigger qui affiche le nombre d'employés insérés dans la table EMPLOYES après chaque insertion.
- Modifier le trigger précédent pour que le déclencheur refuse l'insertion d'un employé dont le salaire est inférieur à 2500 ou supérieur à 20 000 en annulant la transaction et en levant une erreur, sinon il enregistre définitivement la transaction.

Exercices (Correction)

```
-- Trigger afichage après insertion
create or replace trigger Affichage after insert on Empolyes
Begin
raise_application_error (-20000,'Erreur');
End ;
/
```

```
-- Trigger afichage après insertion
create or replace trigger ControleSalaire before update on Empolyes
Begin
if ( :new.salary< 2500 or :new.salary> 20 000 ) then
raise_application_error (-20555,'pas de modification');
End ;
/
```