# Transactions : Ce qu'il faut savoir pour démarrer

## 1. Principes généraux autour de la transaction

Les serveurs de données relationnels fonctionnent en premier lieu comme des moteurs transactionnels (OLTP OnLine Transaction Processing). Ainsi, ces serveurs permettent, à de nombreux usagers d'effectuer de multiples transactions (constituées de une à plusieurs instructions), qui vont être suivies immédiatement d'effets dans un environnement concurrent. Le système doit rester dans un état stable et la notion de transaction va permettre de garantir cette stabilité.

### 1.1 Définition transaction

Une transaction est une unité de traitement séquentiel (séquence cohérente d'actions ), exécutée pour le compte d'un usager, qui appliqée à une base de données cohérente, restitue une base de données cohérente. Les opérations de la transaction doivent être soit exécutées entièrement, soit pas du tout, nous amenant à définir le début et la fin d'un transaction.

Début de la transaction Il est défini de manière implicite par la première commande SQL exécutée ou par la fin d'une transaction précédente (par annulation ou validation de cette dernière).

Fin de la transaction
Elle est soit implicite, soit explicite :

- fin explicite d'une transaction à l'aide des commandes COMMIT (validation des opérations élémentaires) et ROLLBACK (annulation des opérations élémentaires))
- fin implicite d'une transaction
  - exécution d'une commande de définition de données (CREATE, ALTER, RENAME et DROP) : toutes les opérations exécutées depuis le début de la transaction sont validées
  - fin normale d'une session ou d'un programme avec déconnexion d'Oracle : la transaction est validée
  - fin anormale d'un programme ou d'une session (sortie sans déconnexion d'Oracle) : la transaction est annulée

### 1.2 Propriétés d'une transaction

Pour partager en bonne entente, un système multi-concurrent, chacune des transactions doit posséder les propriétés suivantes :

- **Atomicité** : lors d'une exécution d'une transaction, toutes ses actions sont exécutées ou bien aucune ne l'est.
- **Cohérence** : les modifications apportées à la BD lors d'une transaction doivent être valides c'est à dire respecter les contraintes d'intégrité.
- **Isolation** : chaque transaction est isolée, de manière à éviter des incohérences lors d'exécutions concurrentes

— **Durabilité ou Permanence** : les effets d'une transaction qui s'est exécutée correctement doivent survivre à une panne

### 1.3 Modèle général d'une transaction

Le modèle général de la transaction sera le suivant :
Tdébut
—
Actions isolation, atomicité (panne=>défaire)
—
Tfin
Calcul de la validité de la transaction - certification
Point de validation (commit)
Permanence (panne =>refaire éventuellement)
Vrai fin de transaction

Ce que l'on appelle le point de validation (en anglais commit) est fixé pour toute transaction.

— avant un point de validation, une panne entraîne la perte (totale à cause de l'atomicité) de la transaction,
— après le point de validation, la transaction sera visible (au bout d'un certain temps) par les autres.

Cette méthode s'appelle la **validation à deux phases.** Elle suppose souvent l'existence d'une mémoire stable, dans laquelle au point de validation, les nouvelles valeurs devront être enregistrées.

L'isolation d'une transaction passe souvent par la mise en œuvre de verrous (lock en anglais). Les verrous sont dit bloquants dès lors qu'il s'agit d'une opération en écriture. Vous pourrez donc tester les effets d'un ordre UPDATE ou DELETE sur un objet en cours de partage. Vous pourrez aussi exploiter l'ordre SELECT ... FROM ... FOR UPDATE.

## 2. Manipuler la notion de transaction sous Oracle

### 2.1 Niveaux d'isolation

Oracle adopte le principe de modèle de contrôle de concurrence multi-versions ou MVCC (Multi-version Concurrency Control). Dans ce contexte, trois niveaux d'isolation sont disponibles

1. read committed ou read write (mode par défaut : seules les effets des transactions concurrentes validées sont rendues visibles aux autres transactions)
2. read only (aucun action en écriture n'est permise et les effets des transactions concurrentes même validées ne sont pas visbiles)
3. serializable (comme read only mais avec possibilité d'action en écriture)

### 2.2 Pour changer de niveaux d'isolation

Il est possible d'effectuer ces modifications au niveau de la transaction ou au niveau plus global de la session (suite de transactions).

```
SET TRANSACTION READ ONLY NAME 'TransactionUn';
SET TRANSACTION READ WRITE;
```

```
SET TRANSACTION ISOLATION LEVEL READ COMMITTED;
SET TRANSACTION ISOLATION LEVEL SERIALIZABLE;
ALTER SESSION SET ISOLATION_LEVEL = SERIALIZABLE;
ALTER SESSION SET ISOLATION_LEVEL = READ COMMITTED;
```

# 3. Obtenir des privilèges sur d'autres schémas utilisateur

## 3.1 Manipuler GRANT et REVOKE

Ceci est un rappel des ordres GRANT (accorder des droits) et REVOKE (supprimer des droits) qui vous permettront de travailler à plusieurs sur les mêmes objets (tables et tuples) et de vous mettre ainsi en concurrence

```
GRANT [privilege|privilege list|ALL|EXECUTE] ON [object]
TO [schema] [WITH GRANT OPTION];

Exemples :
GRANT INSERT, UPDATE ON Compte TO user1;
GRANT ALL ON Compte  TO public;
GRANT EXECUTE ON f_tranfert TO public;

Syntaxe :
REVOKE [privilege] ([column]) ON [table]
FROM [schema] [CASCADE CONSTRAINTS];

Exemples :
REVOKE UPDATE (solde) ON Compte FROM user3;
```

## 3.2 Visualiser les privilèges

Différentes vues (préfixes USER, ALL et DBA) permettent d'avoir l'information sur les usagers et leurs privilèges concernant les schémas utilisateur.

```
Vues : user_tab_privs
       user_tab_privs_made
       user_tab_privs_recd

desc user_tab_privs

SELECT grantee, table_name, grantor, privilege
FROM user_tab_privs;
```