Notion de transaction

- Une transaction est un ensemble d'ordres SQL qui ont pour objectifs de faire passer la base de données, en une seule étape d'un état cohérent à un autre état cohérent.
- La transaction comporte un début, une suite d'ordres SQL et une fin.
- Si la transaction échoue les modifications déjà effectuées sont annulées : ROLLBACK.
- Si l'ensemble des requêtes peuvent s'exécuter la transaction est validée : COMMIT

Exemple

- Transfert bancaire :
  - débit du compteA <-> crédit du compteB

    Begin transaction

    compteA = compteA – 10

    compteB = compteB + 10

    Commit

    End transaction

Le gestionnaire de transactions

- Il doit assurer le contrôle des transactions et assure les propriétés ACID :
  - Atomicité : Tout ou rien
  - Cohérence : Satisfaction des contraintes d'intégrité
  - Isolation : Pas d'interférences entre les transactions
  - Durabilité : Permanence des modifications effectuées

L'atomicité

- Exécution en tout (COMMIT) ou rien (ROLLBACK)
- La propriété d'atomicité est respectée par le schéma d'exécution suivant :
  - Début transaction
    - Action1, action2, action3 ...
    - Si toutes les actions sont correctement exécutées
    - Alors Valider la transaction (COMMIT)
    - Sinon Annuler ses effets (ROLLBACK)
    - Fin Si
  - Fin transaction

La cohérence

- La cohérence satisfait les contraintes d'intégrité
- Une transaction doit faire passée la base d'un état correct à un autre état correct
- L'ordre des opérations d'une transaction est important
- Exécuter les opérations dans un autre ordre peut conduire à des situations ne vérifiant plus les contraintes d'intégrité

L'isolation

- Les transactions ne doivent pas interférer

T1	T2
– 1 : x <-- lire(X)	– 1 : y <-- lire(X)
– 2 : x <-- x - N	– 2 : y <-- y + M
– 3 : X<-- ecrire(x)	– 3 : X<-- ecrire(y)

- Si l'ordre des transactions est T11, T12, T21, T22, T13, T23 l'effet de l'opération est perdue

La durabilité

- Permanence des modifications effectuées
- Cette permanence doit être vérifiée en cas de panne matérielle ou logicielle
- Le fait que des modifications de données par une transaction ne soit par répercutées dans la base avant leurs utilisations par une autre transaction peut occasionner des problèmes
- Les fichiers journaux permettent de palier au non respect de cette propriété

Concurrence des transactions

- Le SGBD permet à plusieurs transactions de s'exécuter parallèlement
- Chaque « transaction » doit avoir l'impression d'être la seule à utiliser la base
- Le SGBD cherche à maximiser le nombre de transactions et à minimiser les temps d'exécution

Notion d'ordonnancement

- Le processeur partage son temps entre les différentes transactions
- Les lectures/écritures des différentes transactions en cours peuvent être entrelacées
- Les opérations de lecture/écritures peuvent provoquées des interférences si elles s'intéressent aux mêmes objets
- La solution est la sérialisabilité : un ordonnancement est correct s'il est équivalent à au moins un ordonnancement séquentiel des mêmes transactions.

Le verrouillage

- Une transaction verrouille l'objet obligeant les autres transactions à attendre que l'objet se libère
- L'objet ou granule est l'unité verrouillée. Il peut s'agir selon le cas de la base de données, d'une relation, d'un tuple ou d'un attribut.
- Le verrou peut-être binaire, apposé sur un granule, rendant celui-ci accessible ou inaccessible

Le verrouillage (suite)

- Le verrou peut-être de type partagé
  - Share : permet aux autres transactions de lire la donnée (select)
  - Exclusive : la transaction détient le granule pour modifier la donnée (update, insert, delete)

Demande d'une transaction

Verrou apposé

	Share	**Exclusive**
Share	**Oui**	**Non**
Exclusive	**Non**	**Non**

Verrouillage deux-phases (V2P)

- Dans un premier temps la transaction ne peut que verrouiller les données dont elle à besoin, dans un second temps elle ne peut que les déverrouiller
- Pour éviter la situation d'interblocage, si la transaction n'arrive pas à verrouiller l'ensemble des données dont elle à besoin elle doit les déverrouiller

Méthode d'estampillage

- Estampille : identifiant qui définit l'ordre des événements
- Toute transaction reçoit une estampille qui représente le moment où elle commence
- Chaque objet conserve les estampilles read et write de la dernière transaction
- Une transaction ne peut lire ou écrire un objet que si son estampille est supérieure à l'estampille correspondante de l'objet

Segments d'annulation

- Appelés également rollback segments
- Contiennent les anciennes valeurs des enregistrements en cours de modification dans la transaction.
- Ces segments sont utilisés :
  - Pour assurer une lecture consistante des données
  - Pour annuler des transactions
  - En cas de restauration

Les lectures consistantes

L'objectif est de réduire l'attente lors d'accès concurrents.

Oracle assure :

- Qu'au sein d'un ordre SQL les données ne changeront pas
- Les lectures ne seront pas bloquées par des utilisateurs effectuant des modifications
- Les modifications ne seront pas bloquées par utilisateurs effectuant des lectures
- Un utilisateur ne peut lire les modifications effectuées par un autre si elles n'ont pas été validées

Les lectures consistantes (suite)

Oracle conserve toujours les anciennes valeurs et les données mises à jour.

Les anciennes données sont copiées dans les rollback segments avant que les mises à jour ne soient faites dans les segments de données.

Ce n'est qu'au moment du COMMIT que le rollback segment est libéré

S'il y a ROLLBACK les informations du segment d'annulation sont copiées dans les enregistrements affectés

Segment d'annulation (suite)

- Un segment d'annulation est partagée par un certain nombre de transactions
- L'entête d'un segment d'annulation contient une table des transactions ayant des entrées dans le segment
- Les écritures dans les rollback segments sont également journalisés (redo logs)

Segment d'annulation (suite)

- Un rollback segment est un espace physique contenant des rollback entries
- Chaque entrée contient :
  - L'identification du fichier de données
  - L'identification du bloc contenant les données modifiées
  - La valeur avant modification de la donnée
- Les entrées correspondant à une même transaction sont chaînées entre elles

Types de tables MySQL

- MySQL supporte plusieurs moteurs de stockage, qui gère différents types de tables.
- Les moteurs de tables MySQL peuvent être transactionnels ou non-transactionnels.
- Lorsque vous créez une table, vous pouvez indiquer à MySQL le type de table avec la clause ENGINE

```
CREATE TABLE t (i INT) ENGINE = INNODB;
```

- Si vous omettez ENGINE, le type de table par défaut sera utilisé. C'est généralement MyISAM. Cela peut être changé en modifiant la variable système table_type.

Moteurs non-transactionnels

- Le moteur de tables originel était ISAM, qui gérait des tables non-transactionnelles. Ce moteur a été remplacé par le moteur MyISAM
- Le moteur HEAP propose des tables stockées en mémoire
- Le moteur MERGE permet le regroupement de tables MyISAM identiques sous la forme d'une seule table

Moteurs transactionnels

- Les moteurs InnoDB et BDB gèrent des tables transactionnelles
- NDBCluster est le moteur de stockage du cluster MySQL qui implémente des tables réparties sur plusieurs serveurs.

Création de tables MySQl

- MySQL crée toujours un fichier .frm pour stocker le type de la table et les informations de définition.
- Le serveur crée le fichier .frm par dessus le moteur de stockage.
- Les moteurs peuvent créer des fichiers supplémentaires, en fonction de leurs besoins.

Les avantages des tables transactionnelles

- Plus sûr. Même si MySQL crashe ou que vous avez un problème matériel, vous pouvez récupérer vos données, soit par un recouvrement automatique, soit à partir d'une sauvegarde combinée avec le log des transactions.
- Vous pouvez combiner plusieurs commandes et les accepter toutes d'un seul coup avec la commande COMMIT.
- Vous pouvez utiliser ROLLBACK pour ignorer vos modifications (si vous n'êtes pas en mode auto-commit).

Les avantages des tables transactionnelles (suite)

- Si une mise à jour échoue, tout vos changements seront annulés. (Avec les tables non-transactionnelles tous les changements opérés sont permanents)
- Gère mieux les accès concurrents si la table reçoit simultanément plusieurs lectures.

Avantages des tables non-transactionnelles

- Plus rapides
- Utilisent moins d'espace disque
- Utilisent moins de mémoire pour exécuter les mises à jour.

Type de table Sakila

```
mysql> select table_name, engine
    -> from information_schema.tables
    -> where table_schema='sakila'
    -> and table_type='base table';
```

```
+---------------+--------+
| table_name    | engine |
+---------------+--------+
| actor         | InnoDB |
| address       | InnoDB |
| category      | InnoDB |
| city          | InnoDB |
| country       | InnoDB |
| customer      | InnoDB |
| employe       | InnoDB |
| film          | InnoDB |
| film_actor    | InnoDB |
| film_category | InnoDB |
| film_text     | MyISAM |
| inventory     | InnoDB |
| inventory_sav | InnoDB |
| language      | InnoDB |
| payment       | InnoDB |
| rental        | InnoDB |
| staff         | InnoDB |
| store         | InnoDB |
+---------------+--------+
```

Présentation des tables innoDB

- Gestionnaire de table transactionnelle (compatible ACID), avec validation (commits), annulations (rollback) et capacités de restauration après crash.
- Utilise un verrouillage de lignes, et fournit des lectures cohérentes comme Oracle, sans verrous.