Transaction SQL

Par défaut MySQL est en mode AUTOCOMMIT=1, chaque requête est validée directement l'une à la suite de l'autre. Donc, si dans l'enchaînement de plusieurs requêtes une échoue, par exemple, certaine(s) peuve(nt) déjà avoir modifiée(s) les tables de base de données. La base de données devient incohérente d'un point de vue de l'information quelle contient.
Le système des transactions permet de valider les requêtes si tout c'est bien passé (notion Atomique, une transaction un atome...voir plus loin).
Pour valider ou annuler une requête on commit ou on fait un rollback.

L'exemple que l'on prend souvent pour décrire l'utilité des transactions c'est une personne qui achète un produit sur Internet et qui est débité d'un certain montant puis, ce montant crédite une autre table vendeur, théoriquement, mais, cette étape peut par exemple ne pas marcher pour une raison ou une autre. Sans transaction la personne est débitée mais ne reçoit jamais son colis puisque le vendeur n'a jamais été payé, c'est donc la catastrophe...En fait il faut voir ces requêtes comme une et une seule chose : une transaction !

Plus précisément :
Les transactions c'est un ensemble de requêtes qui sont exécutées en un seul bloc, ainsi si une requête échoue alors on peut décider d'annuler 'rollback' toutes les requêtes, sorte de CTRL+Z pour les commandes SQL ; ou bien si tout réussi on valide 'commit' le bloc de requêtes.

Attention, tous les moteurs ne supportent pas les transactions, le moteur InnoDB est transactionnel pas MyISAM par exemple.

Par défaut chaque requête de MySQL est validité, elle est commiter.

Si on veut modifier cet état de MySQL on écrit mais, cette méthode n'est pas conseillée :

SET AUTO_COMMIT=0 ; ← la valeur par défaut est 1 tout commit ...

Dans ce cas il faudra « commiter » les requêtes pour qu'elles soient effectives dans la base de données.

COMMIT ;

Pour annuler une requête :

ROLLBACK ;

On utilise plutôt START TRANSACTION pour commencer une transaction, c'est plus propre et cela ne dure que le temps de la transaction, c'est-à-dire quand la commande COMMIT ou ROLLBACK est lancée.

INSERT INTO article (title) VALUES ('un titre') ;

START TRANSACTION

un COMMIT ou RALLBACK met fin à la transaction, MySQL revient en mode AUTO_COMMIT=0
STRART TRANSACTION ← début

COMMIT ← fin de la transaction

Exemple :

```
START TRANSACTION;
INSERT INTO article (title, content)
VALUES ('Hello','je suis un contenu');
SAVEPOINT jalon1; ← permet d'annuler à partir de c'est jalon
INSERT INTO commen (title, content)
VALUES ('super titre', 'je suis un commentaire');
ROLLBACK TO SAVEPOINT jalon1; ← annulation des commandes depuis le jalon1
INSERT INTO category (title)
VALUES ('linux');
COMMIT;
```

Attention toutes les commandes DDL valide les transactions, les commandes qui influent sur la structure de la base de données en générale provoque la validation implicite.

ACID

Une transaction doit être atomique elle doit former une entité complète et indivisible. En clair chaque élément de la transaction ne peut exister en dehors de la transaction elle-même.
Un virement bancaire est constitué de deux actions indivisibles. C'est l'**atomicité.**

Les données restent cohérente dans tous les cas, que la transaction se termine ou pas (rollback ou commit)

De plus chaque transaction ne doit pas interagir avec une autre...

Isoler les transactions le sont si on a pas commité dans une session les transactions en cours, une deuxième session ne pourra pas modifier des données qui sont en cours de modification dans la première session. Il y a un verrou qui est posé sur les lignes affectées. On est en mode multi-utilisateur...

Durabilité les données stockées sont gardées en base de données.