# Bases de Données : Transactions

Stéphane Devismes

Université Grenoble Alpes

26 août 2020





## Plan







## Consulter *vs* modifier

Trois niveaux d'intervenants :

- L'utilisateur peut seulement *consulter* les données. **(requêtes)**
- Le gestionnaire peut *consulter* et *modifier les données*. **(Insérer, supprimer, modifier des lignes dans les tables)**
- L'informaticien peut *consulter*, *modifier les données* et *modifier les tables*.

[4ème niveaux : la gestion des droits ou « privilèges » avec le DCL *Data Control Language*, langage de contrôle des données (HORS PROGRAMME).]





## Modifications des données

Aujourd'hui, on considère **les modifications des valeurs** (*i.e.*, des lignes) MAIS pas les modifications de la structure (*i.e.* des tables).

Les modifications des données nécessitent plus de soin que les requêtes, afin de ne pas « abîmer » la base de données : dans ce but, on utilise des transactions.

## Transactions

Une transaction est une suite cohérente de consultations et/ou modifications d'une base de donnée.

Une transaction doit être soit **totalement écrite**, soit **totalement annulée**.

Une transaction est atomique : les opérations d'une transaction sont *toutes* exécutées, sans qu'aucune *autre* opération (d'une autre transaction) s'intercale.





## Ecrire totalement ou annuler totalement en SQL

- L'ordre SQL COMMIT termine et valide (écrit) la transaction.
- L'ordre SQL ROLLBACK termine et annule la transaction.

Il faut bien faire la différence entre :

- effectuer une modification (elle est faite, mais elle peut être annulée par un simple ROLLBACK),
- confirmer cette modification (par COMMIT) : ensuite elle ne peut plus être annulée par un simple ROLLBACK.





## COMMIT implicite

Certains ordres SQL **COMMITent** automatiquement.

Par exemple, tous ceux qui ne portent pas directement sur le contenu des données : CREATE, DROP, ALTER, GRANT, REVOKE, *etc.*





## ROLLBACK implicite

Un mécanisme de sécurité et d'intégrité basique d'Oracle fait qu'une transaction est automatiquement annulée en cas de défaillance soit du poste client, soit du serveur, voire du réseau.

Par exemple, CTRL + ALT + DEL du PC client

## COMMIT et ROLLBACK implicites : résumé

| Validation de transaction | Annulation de transaction |
|---|---|
| COMMIT | ROLLBACK |
| ordres SQL : | interruption : |
| CREATE, DROP, ALTER | CTRL ALT DEL, RESET, ON/OFF |
| ordre SQL*Plus : EXIT, QUIT | erreur d'exécution Oracle |





## AUTOCOMMIT

Il y a dans Oracle une option AUTOCOMMIT.

Par défaut, AUTOCOMMIT est OFF (SHOW AUTOCOMMIT répond OFF).

On peut mettre cette option à ON (par SET AUTOCOMMIT ON, ensuite SHOW AUTOCOMMIT répond ON).

Alors chaque opération SQL est considérée comme une transaction et elle est confirmée immédiatement après avoir été effectuée.





## Sous-transactions

Il est possible de définir des sous-transactions et de faire des **annulations partielles** de transaction jusqu'à une **étiquette** préalablement définie :

```
UPDATE EMP SET SAL=...
SAVEPOINT S1
UPDATE EMP SET COMM=...
ROLLBACK TO S1
```





## Remarques

Une modification des valeurs d'une base n'est autorisée que si elle respecte les contraintes de la base.

Une modification des valeurs concerne une table en mémoire, jamais une vue.

## Modifications possibles

Ajout d'une ou plusieurs lignes :

```
INSERT INTO nomTable (listeColonnes) VALUES (listeValeurs)

INSERT INTO nomTable (listeColonnes) SELECT...
```

Mise à jour de la valeur d'un ou plusieurs attributs :

```
UPDATE nomTable
SET colonne1 = valeur1, colonne2 = valeur2, ...
[WHERE condition]
```

Suppression d'une ou plusieurs lignes.

```
DELETE FROM nomTable [WHERE condition]
```





## Ajout : exemple (2/2)

Inscrire en athlétisme tous les élèves qui ne sont inscrits à aucune activité.





## Ajout : exemple (1/2)

Ajouter le nouvel élève Xavier Leblanc, né le 30 juin 1988, dans l'option 2.



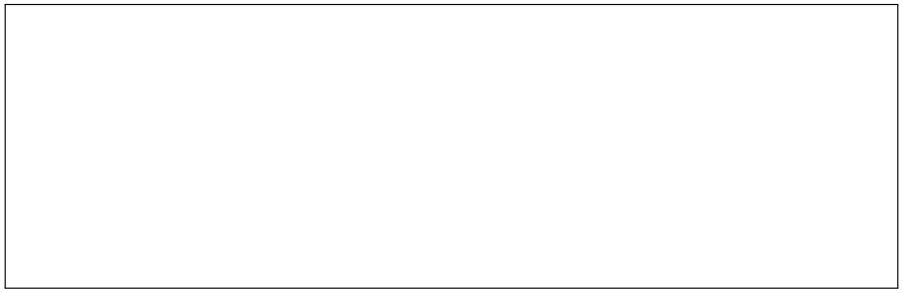


## Mise à jour et suppression : exemple (1/2)

On suppose que l'attribut `opt` de la table `Eleves` peut être `NULL`.

1. L'élève Michel Leblanc passe de l'option 2 à l'option 1.
2. Désinscrire de l'option 1 l'élève Michel Leblanc, mais le laisser inscrit à l'école.
3. Désinscrire l'élève Michel Leblanc de toutes ses activités.
4. Revenir à la situation initiale.

# Mise à jour et suppression : exemple (2/2)







# Valeurs absentes : exemple (1/3)

≪ NULL ≫ se comporte comme les autres valeurs : on peut utiliser ≪ NULL ≫ à la place d'une autre valeur dans un ajout, une modification, une suppression.

Dans les conditions, comme pour les requêtes, on doit utiliser `IS NULL` ou `IS NOT NULL`.





# Valeurs absentes : exemple (2/3)

```
CREATE TABLE Test (
  nb   NUMBER(10),
  st   VARCHAR2(20)
);
-- Table created.
```





# Valeurs absentes : exemple (3/3)

```
INSERT INTO Test (nb, st) VALUES (1, NULL);
INSERT INTO Test (nb, st) VALUES (NULL, 'A');

SELECT * FROM Test;
-- 2 rows selected (1.null, null.A)

UPDATE Test SET nb = 2 WHERE st IS NULL;

SELECT * FROM Test;
-- 2 rows selected (2.null, null.A)

UPDATE Test SET st = 'B' WHERE st IS NOT NULL;

SELECT * FROM Test;
-- 2 rows selected (2.null, null.B)

UPDATE Test SET nb = NULL WHERE st IS NULL;

SELECT * FROM Test;
-- 2 rows selected (null.null, null.B)

DELETE FROM Test WHERE nb IS NULL;
-- 2 rows deleted.

SELECT * FROM Test;
-- no rows selected

DROP TABLE Test;
-- Table dropped.
```

## Quatre propriétés essentielles

- Atomicité
- Cohérence
- Isolation
- Durabilité





## Atomicité

Une transaction doit soit être complètement validée ou complètement annulée : c'est la règle du « tout-ou-rien » !

Par exemple lors d'une opération informatique de virement d'un compte bancaire $c$ sur un autre compte bancaire $c'$, il y a une opération de retrait sur $c$ et une opération de dépôt sur $c'$.

La transaction regroupe ces deux opérations.

Si elles sont toutes les deux effectuées sans erreur, alors la modification est confirmée (elle devient effective) sur les deux comptes (on parle alors de `COMMIT`).

Si ce n'est pas le cas la transaction est annulée, les deux comptes gardent leur valeur initiale (on parle alors de `ROLLBACK`).





## Cohérence

Une transaction doit laisser la base de données dans un état cohérent.

Cela signifie que les contraintes de la base de données doivent toujours être satisfaites.

En fait, le système vérifie les contraintes à chaque modification, donc a fortiori elles sont satisfaites à chaque transaction.





## Isolation

Une transaction ne peut voir aucune autre transaction en cours d'exécution.

Même si plusieurs transactions sont exécutées en même temps, le système doit mettre en place un mécanisme pour faire comme si elles étaient exécutées l'une après l'autre : sérialisation, verrous (« locks »), ordonnancement (« scheduling ») . . .

Par exemple lors d'une réservation de billets de train par internet la même place dans le même train ne doit pas être attribuée à deux personnes différentes.

## Durabilité

Après que le client a été informé du succès de la transaction, les résultats de celle-ci ne doivent pas disparaître.

Dans l'exemple d'un virement d'un compte bancaire $c$ sur un autre compte bancaire $c'$, si les deux opérations (retrait de $c$ et dépôt sur $c'$) sont effectuées, et si le client est informé du succès de la transaction dès que celle-ci est dans la liste d'attente du disque, sans attendre qu'elle soit réellement prise en compte dans la base, alors en cas de panne de courant la durabilité n'est pas satisfaite.





## ≪ petit monde ≫

Les propriétés précédentes sont valables sous l'hypothèse de ≪ petit monde ≫ : on peut empêcher l'accès à la base de données pendant qu'on enregistre la transaction.

D'autres idées doivent être développées pour internet et le ≪ cloud ≫, ces idées sont souvent regroupées sous le mot NOSQL, qui signifie "Not Only SQL".





## Modification des tables

L'informaticien peut (aussi) modifier les tables, c'est-à-dire qu'il est autorisé à créer et à modifier le schéma et la spécification des tables.

Pour ces modifications, les transactions sont commitées implicitement.