

# Leçon 1 : LE LANGAGE DE CONTRÔLE DE DONNÉES

Université Virtuelle de Côte d'Ivoire



# Table des matières

<b>I - 1- Généralité sur le langage de contrôle de données</b>	<b>3</b>
<b>II - Application 1 :</b>	<b>4</b>
<b>III - 2- Gestions des transactions</b>	<b>5</b>
<b>IV - Application 2 :</b>	<b>7</b>
<b>V - 3- Privilèges et Utilisateurs</b>	<b>8</b>
<b>VI - Application 3</b>	<b>11</b>
<b>Solutions des exercices</b>	<b>12</b>

# 1- Généralité sur le langage de contrôle de données



## 1.1- Définition

Un langage de contrôle de données (LCD ) ou *data control language (DCL)* en anglais est un langage de programmation et un sous-ensemble de SQL pour contrôler l'accès aux données d'une base de données.

Elle consiste à gérer les permissions et annulations des transactions.

## 1.2- Les Commande SQL

Les commandes SQL utilisées dans le langage de contrôle de données sont :

*GRANT* : autorisation d'un utilisateur à effectuer une action ;

*DENY* : interdiction à un utilisateur d'effectuer une action ;

*REVOKE* : Supprime des autorisations ;

*COMMIT* : validation d'une transaction en cours ;

*ROLLBACK* : annulation d'une transaction en cours ;

*LOCK* : verrouillage sur une structure de données.

# Application 1 :



## Exercice

[solution n°1 p.12]

Le langage de contrôle de données est :

- ☐ un langage d'analyse de données
- ☐ un langage de contrôle des intrusions
- ☐ un langage pour contrôle l'accès aux données

## Exercice

[solution n°2 p.12]

L' instruction de validation d'une transaction en cours en SQL est :

L' instruction d'autorisation d'un utilisateur à effectuer une action en SQL est :

# 2- Gestions des transactions



## 2.1- Définition

Une transaction, c'est un ensemble de requêtes qui sont exécutées en un seul bloc. Ainsi, si une des requêtes du bloc échoue, on peut décider d'annuler tout le bloc de requêtes (ou de quand même valider les requêtes qui ont réussi).

*Exemple de transaction :*

M. Koffi effectue un transfert d'argent d'un compte bancaire vers autre compte. Pendant le processus, une panne survient et le compte de M. Koffi a été débité de la somme de 500.000FCFA sans que l'autre compte soit crédité du même montant.

*Le mécanisme transactionnel* empêche un tel scénario en invalidant toutes les opérations faites depuis le début de la transaction, si une panne survient au cours de cette même transaction.

## 2.2- Syntaxe création d'une transaction


*START TRANSACTION*

*< Bloc de requête >*

*COMMIT OU ROLLBACK*

*NB :*

- *Démarre une transaction : START TRANSACTION*
- *< Bloc de requête > : il s'agit de toutes les requêtes en SQL*
- *Valider les modifications : COMMIT*
- *Annuler les modifications : ROLLBACK*
- *Possibilité de mettre des points de reprise : SAVEPOINT*

 *Remarque : Remarque :*

---

*Nos exemples seront implémentés avec MySQL.*

*NB :*

MySQL ne travaille pas avec les transactions. Chaque requête effectuée est directement commitée (validée). On ne peut pas revenir en arrière. On peut donc en fait considérer que chaque requête constitue une transaction, qui est automatiquement commitée. Par défaut, MySQL est donc en mode "autocommit".

Pour quitter ce mode, il suffit de lancer la requête suivante :

*SET autocommit=0;*



### *Exemple : Exemple :*

Supposant qu'un client « Koffi » souhaite transférer un montant de 500.000 FCFA vers le compte de son fournisseur « industrie ». L'opération sera effectuée à travers la banque en transférant le montant de 500.000 FCFA du compte « Koffi » au compte « Industrie ».

Voici la liste des ordres SQL à exécuter :

```
1 SET autocommit = 0 ; //----Quitter le mode Autocommit de mysql
2 START TRANSACTION ; //----- Début des la transaction
3 update compte set montant = montant - 500000 where nom='Koffi';
  //soustraire les 500000 du compte de koffi sans valider
4 update compte set montant = montant + 500000 where nom='Industrie';
  //Ajoute les 500000 sur le compte d'industrie sans valider
5 COMMIT ; //---Confirmer la transaction en validant les requêtes précédentes
```

# Application 2 :

IV

## Exercice

[solution n°3 p.12]

Énoncé :

Réaliser une transaction qui enregistre 3 clients avec les propriétés suivantes : identifiant,nom, prenom,datenaissance. Modifier ensuite la date de naissance du client 03 à 18-02-1980, cette modification ne doit pas être pris en compte par la transaction.

NB : Respecter l'ordre de création des propriétés

Solution :

```

[ ] [ ]
[ ] [ ] clients( [ ] )
VALUES ('01', 'KOFFI', 'GERARD', '1986-02-15');
[ ] [ ] clients( [ ] )
VALUES ('02', 'KOFFI', 'INES', '1988-06-11');
[ ] [ ] clients( [ ] )
VALUES ('03', 'KONE', 'ISSA', '1976-07-10');
[ ]
[ ] clients
SET [ ]
[ ] [ ]
[ ]

```

# 3- Privilèges et Utilisateurs



## 3.1- Définition

Un privilège est le droit d'exécuter une requête SQL spécifique attribué à des utilisateurs.

Chaque utilisateur possède une série de privilèges, relatifs aux données stockées sur le serveur : le privilège de sélectionner les données, de les modifier, de créer des objets, etc.

Ces privilèges peuvent exister à plusieurs niveaux: base de données, tables, colonnes, procédures, etc.

## 3.2- Création et suppression des utilisateurs

- *Syntaxe de Création :*

```
CREATE USER 'login'@'hote' IDENTIFIED BY 'mot_de_passe';
```

*login* est un simple identifiant

*hôte* est l'adresse à partir de laquelle l'utilisateur va se connecter (localhost pour une adresse locale)

*mot\_de\_passe* : le mot de passe pour se connecter

*Exemple :*

```
CREATE USER 'ut1'@'localhost' IDENTIFIED BY 'ut12019';
```

```
CREATE USER 'ut2'@'192.28.12.4' IDENTIFIED BY 'ut22019';
```

```
CREATE USER 'ut3'@'uvci.edu.ci' IDENTIFIED BY 'ut32019';
```

- *Syntaxe de Suppression*

```
DROP USER 'login'@'hote';
```

*Exemple :*

```
DROP USER 'ut3'@'uvci.edu.ci';
```

```
DROP USER 'ut2'@'192.28.12.4';
```

```
DROP USER 'ut1'@'localhost';
```

## 3.3- Attribuer et retirer un privilège

Il existe de nombreux privilèges dont voici une sélection des plus utilisés sont :



- *Privilèges du CRUD*

les privilèges *SELECT*, *INSERT*, *UPDATE* et *DELETE* permettent aux utilisateurs d'utiliser ces mêmes commandes.

- *Privilèges concernant les tables, les vues et les bases de données*

Privilège	Action autorisée
<i>CREATE TABLE</i>	Création de tables
<i>CREATE TEMPORARY TABLE</i>	Création de tables temporaires
<i>CREATE VIEW</i>	Création de vues (il faut également avoir le privilège <i>SELECT</i> sur les colonnes sélectionnées par la vue)
<i>ALTER</i>	Modification de tables (avec <i>ALTER TABLE</i> )
<i>DROP</i>	Suppression de tables, vues et bases de données
<i>CREATE ROUTINE</i>	Création de procédures stockées (et de fonctions stockées - non couvert dans ce cours)
<i>ALTER ROUTINE</i>	Modification et suppression de procédures stockées (et fonctions stockées)
<i>EXECUTE</i>	Exécution de procédures stockées (et fonctions stockées)
<i>INDEX</i>	Création et suppression d'index
<i>TRIGGER</i>	Création et suppression de triggers
<i>CREATE USER</i>	Gestion d'utilisateur (commandes <i>CREATE USER</i> , <i>DROP USER</i> , <i>RENAME USER</i> et <i>SET PASSWORD</i> )

- *Attribuer un privilège*

*GRANT* privilege [(liste\_colonnes)] [, privilege [(liste\_colonnes)]

*ON* [type\_objet] niveau\_privilege

*TO* utilisateur [*IDENTIFIED BY* mot\_de\_passe];

*privilege* : le privilège à accorder à l'utilisateur (*SELECT*, *CREATE VIEW*, *EXECUTE*,...);

(*liste\_colonnes*) : facultatif - liste des colonnes auxquelles le privilège s'applique ;

*niveau\_privilege* : niveau auquel le privilège s'applique (\*.\*, nom\_bdd.nom\_table,...) ;

*type\_objet* : en cas de noms ambigus, il est possible de préciser à quoi se rapporte le niveau : TABLE ou PROCEDURE.

*Example :*

Attribuer les privilèges SELECT,UPDATE,DELETE , INSERT et UPDATE ( pour les champs nom, prenom, datenaiss) pour la base de données gestion\_ecole sur la table etudiant

```
GRANT SELECT,
UPDATE (nom, prenom, datenaiss),
DELETE,
INSERT
ON gestion_ecole.etudiant
TO 'ut1'@'localhost'
```

- *Révocation de privilège*

Pour retirer un ou plusieurs privilèges à un utilisateur, on utilise la commande *REVOKE*.

*REVOKE* privilege [, privilege, ...]

*ON* niveau\_privilege

*FROM* utilisateur;

*Exemple :*

Retirer le droit de suppression dans la base de données gestion\_ecole pour l'utilisateur ut3.

REVOKE DELETE

ON gestion\_ecole

FROM 'ut3'@'uvci.edu.ci';

# Application 3

VI

## Exercice

[solution n°4 p.13]

Énoncé :

Créer un utilisateur akj sur le serveur local avec un mot de passe akj2019

Attribuer les privilèges CREATE TABLE et DROP à akj dans la base gestion\_client

CREATE

TO



# Solutions des exercices

### > Solution n°1

Exercice p. 4

Le langage de contrôle de données est :

- ☐ un langage d'analyse de données
- ☐ un langage de contrôle des intrusions
- ☒ un langage pour contrôle et l'accès aux données

### > Solution n°2

Exercice p. 4

L' instruction de validation d'une transaction en cours en SQL est : COMMIT

L' instruction d'autorisation d'un utilisateur à effectuer une action en SQL est : GRANT

> **Solution n°3**

Exercice p. 7

Énoncé :

Réaliser une transaction qui enregistre 3 clients avec les propriétés suivantes : identifiant,nom,prenoms, datenaissance. Modifier ensuite la date de naissance du client 03 à 18-02-1980, cette modification ne doit pas être pris en compte par la transaction.

NB : Respecter l'ordre de création des propriétés

Solution :

```
START TRANSACTION
INSERT INTO clients(identifiant,nom,prenoms,datenaissance)
VALUES ('01','KOFFI','GERARD','1986-02-15');
INSERT INTO clients(identifiant,nom,prenoms,datenaissance)
VALUES ('02','KOFFI','INES','1988-06-11');
INSERT INTO clients(identifiant,nom,prenoms,datenaissance)
VALUES ('03','KONE','ISSA','1976-07-10');
COMMIT;
UPDATE clients
SET datenaissance='1980-02-18'
WHERE identifiant='03';
ROLLBACK;
```

> **Solution n°4**

Exercice p. 11

Énoncé :

Créer un utilisateur akj sur le serveur local avec un mot de passe akj2019

```
CREATE USER 'akj'@'localhost' IDENTIFIED BY 'akj2019';
```

Attribuer les privilèges CREATE TABLE et DROP à akj dans la base gestion\_client

```
GRANT
CREATE TABLE,
DROP
ON gestion_client
TO 'akj'@'localhost';
```