BNF101 Base de Données Relationnelles

Création des tables de la BDD COMMANDE

SQL Langage de Manipulation des Données

Insertion des Données

CORRECTION DES EXERCICES

Fichier create_table_oracle_2025_2025.sql et create_table_postgres_2025_2025_v2.sql

⇒Création de la table CLIENT : 1 script de création de la table sans clé primaire + 1 script de modification de la table avec l'ajout de la clé primaire.

⇒Création de la table CLIENT2 : 1 script de création de la table sans clé primaire + 1 script de modification de la table avec l'ajout de la clé primaire + 1 script de création de séquence gérer la clé primaire en auto-incrément.

⇒Création des tables VENDEUR, PRODUIT, COMMANDE, LIGNECOM : 1 script de création par table avec les clés et les contraintes.

+ scripts de modification de colonnes pour chacune de ces tables.

Le choix d'un script de création unique ou d'un script de création et de modification varie en fonction des contraintes d'exploitation de la Base de Données.

COMPARAISON DES SGBDR ORACLE ET POSTGRES

Comparaison des fichiers create_table_oracle_2024_2025.sql et create_table_postgres_2024_2025_v2.sql.

- La syntaxe des scripts SQL de création des tables est presque identique entre ORACLE et POSTGRES.

Cependant, le type NUMBER n'existe pas dans POSTGRES : il est remplacé par le type INTEGER (entier) ou NUMERIC (avec une partie décimale).

- La syntaxe des scripts SQL de modification des colonnes est différente entre ORACLE et POSTGRES.

MODIFY <nom de la colonne> dans **ORACLE** et ALTER COLUMN <nom de la colonne> SET dans **POSTGRES**.

- La syntaxe de création des séquences sur les clés primaires est différente entre ORACLE et POSTGRES.

Dans **ORACLE** on doit explicitement créer une séquence et utiliser la séquence dans les requêtes d'insertion des données.

Dans **POSTGRES**, il suffit d'ajouter le mot clé SERIAL sur la ligne de création de la colonne clé primaire dans le script de création de la table. Et **POSTGRES** crée automatiquement la séquence lors de la création de la table.

LANGAGE DE MANIPULATION DES DONNEES (LMD)

Le langage de manipulation des données (LMD ou DML) est utilisé lorsque l'on souhaite ajouter, mettre à jour ou supprimer des données dans une base de données (BDD).

Le LMD est donc utilisé pour gérer le contenu d'une BDD alors que le LDD est utilisé pour gérer le contenant : les tables, les clés et les contraintes.

LMD : INSERTION DES DONNEES

Insertion de données

Une fois la table créée il s'agit d'y inserer des occurences et ce en utilisant l'instruction INSERT:

INSERT INTO nom_de table **(**col 1,… , col n**) VALUES (** val 1, ..., val n**) ;**

Supposons que nous disposons de la table CLIENT ayant la structure suivante:

CLIENT (NumCli NUMBER(3), NomCli VARCHAR2(30), AdresseCli VARCHAR2(40), codePos NUMBER(5), Ville VARCHAR2(30), Tel NUMBER(10)).

Si nous désirons insérer l'occurence relative au CLIENT numéro 1 nommé 'redford‘ demeurant '8 rue de la paix 75002 paris‘ et dont le numéro de téléphone est le 0142232323.

INSERT INTO CLIENT **(**NumCli, NomCli, AdresseCli, codePos, Ville, Tel**) VALUES (**1,'redford‘ ,'8 rue de la paix’, 75002, ‘paris‘, 0142232323 **);**

Remarques

1) Une chaine de caractère est insérée entre cote simple ' chaine'

2) Il est important d'avoir une correspondance entre l'ordre d'une valeur et l'ordre de la colonne associée.

LMD : INSERTION DES DONNEES

5) Si l'ordre de présentation des valeurs n'est pas le même que celui de la définition ou si on a moins de valeurs que le nombre total des Colonnes, il faut revenir à la première syntaxe en mentionnant les colonnes concernées.

Exemple pour un numéro de téléphone non renseigné pour 1 client :

INSERT INTO CLIENT **(**NumCli, NomCli, AdresseCli, codePos, Ville**) VALUES (**1,'redford‘ ,'8 rue de la paix’, 75002, ‘paris‘**)**

Préconisation concernant les valeurs NULL :

Dans le cas où l’on ne dispose pas de valeur pour une colonne, il préférable d’alimenter la requête dans sa forme complète :

INSERT INTO CLIENT **(**NumCli, NomCli, AdresseCli, codePos, Ville, Tel**) VALUES (**1,'redford‘ ,'8 rue de la paix’, 75002, ‘paris‘, null**)**

LMD : INSERTION DES DONNEES

3) Si on cite tous les noms des colonnes, on n'est pas obligé de se conformer à l'ordre selon lequel les colonnes ont été créées. La requête précédente donne le même résultat que celle qui suit:

INSERT INTO CLIENT **(**NumCli, Tel, AdresseCli, Ville, NomCli , codePos**) VALUES (**1, 0142232323,'8 rue de la paix', 'paris','redford' ,75002**);**

4) La syntaxe de l'instruction INSERT peut être simplifiée dans le cas où on a autant de valeurs que de colonnes et en respectant l'ordre de présentation des colonnes et des valeurs. En effet les deux requêtes précédentes donnent le même résultat que celle qui suit:

INSERT INTO CLIENT **VALUES (**1,'redford' ,'8 rue de la paix', 75002, 'paris', 0142232323**);**

Cette forme est possible car la structure de la table est composée de 6 colonnes et on a exactement une valeur pour chaque colonne. Ces valeurs sont en plus présentées dans l'ordre de création des colonnes.

LMD : NOTION DE TRANSACTION

Une série de requêtes DML forme une unité logique de travail appelée transaction.

Exemple de la BDD commandes :

Lorsque l'on ajoute une commande contenant 2 produits différents, la transaction peut être décomposée en 3 opérations distinctes :

- Ajout de la commande,
- Ajout de la ligne de commande du 1er produit,
- Ajout de la ligne de commande du 2ème produit.

Le serveur oracle doit garantir que chacune des trois requêtes a été exécutée correctement.

Si au moins l'une des trois requêtes a échouée, les autres requêtes doivent être annulées.

LMD : COMMIT ET ROLLBACK

Une transaction débute dès l'exécution de la 1ère requête LMD.

Tout changement de données pendant une transaction est temporaire jusqu'à ce que la transaction soit validée.

Dans le cas de la gestion du contenu, la transaction doit être validée explicitement à l'aide de la commande COMMIT.

Tant que la commande COMMIT n'a pas été exécutée, la transaction peut être annulée à l'aide de la commande ROLLBACK.

Remarque concernant les requêtes DDL (CREATE, ALTER, DROP) pour la gestion du contenant :

la validation des requêtes est implicite (COMMIT implicite ou AUTO COMMIT).

LMD : COMMIT ET ROLLBACK

Remarque sur les sessions SQL PLUS (ORACLE):

Lorsque l'on quitte correctement une session SQL PLUS via la commande « quit », le serveur ORACLE réalise un COMMIT sur les données manipulées lors d'une transaction LMD avant de fermer la session.

En revanche si l'on ne quitte pas correctement la session SQL PLUS, suite à un problème système par exemple, le serveur ORACLE réalise un ROLLBACK sur la transaction.

LMD : COMMIT ET ROLLBACK

Remarque sur la commande AUTOCOMMIT dans SQL*PLUS :

La commande AUTOCOMMIT peut être paramétrée en état ON ou OFF. Si autocommit est activé (grâce à la commande sqlplus SET AUTOCOMMIT ON), chaque requête DML est validée dès son exécution.

=> On ne peut pas annuler les changements avec un ROLLBACK.

Si autocommit est désactivé (état par défaut : SET AUTOCOMMIT OFF), l'ordre COMMIT peut toujours être utilisé de manière explicite.

=> On peut annuler les changements avec un ROLLBACK.

LMD : COMMIT ET ROLLBACK

Remarque sur la gestion du COMMIT dans le SGBD Postgresql :

La directive COMMIT est activée par défaut dans postgresql.

Il n'est donc pas nécessaire de lancer la commande COMMIT après avoir lancé une ou plusieurs requête SQL d'insertion.

Il est possible de valider explicitement une requête d'insertion de données.

Pour ce faire, il faut découper le script en transactions comme suit :

```
begin;
<requêtes sql d'insertion>
Commit;
end transaction;
```

COMMIT ET ROLLBACK : RESUME DES CAS D'USAGE

validation de transaction	**annulation de transaction**
COMMIT	ROLLBACK
ordres SQL DDL (autocommit) : CREATE, DROP, ALTER, RENAME, TRUNCATE	
Ordres SQL DMP (commit explicite) : INSERT, DELETE, UPDATE	Ordres SQL DMP (rollback explicite) : INSERT, DELETE, UPDATE
ordre SQL*Plus : EXIT, QUIT, CONNECT, DISCONNECT	erreur d'exécution Oracle

LMD : COMMIT ET ROLLBACK

État des données avant un COMMIT ou un ROLLBACK

Tout changement de données pendant une transaction est temporaire jusqu'à ce que la transaction soit validée.

L'état des données avant que la commande COMMIT ou ROLLBACK ne soit exécutée se résume comme suit :

- Les requêtes LMD modifient la mémoire tampon de la base de données de telle sorte que l'état précédent des données peut être récupéré.
- L'utilisateur courant peut vérifier les résultats des opérations LMD en interrogeant les tables.
- Les autres utilisateurs ne peuvent pas voir les résultats des requêtes LMD exécutées par l'utilisateur courant,
- Les lignes affectées sont verrouillées, les autres utilisateurs ne peuvent pas changer les données dans les lignes affectées.

Le serveur Oracle fournit une consistance de lecture qui assure que chaque utilisateur voit les données telles qu'elles étaient au moment du dernier COMMIT.

LMD : COMMIT ET ROLLBACK

État des données après un COMMIT

L’exécution de la commande COMMIT valide les changements de données opérés par une transaction.

La séquence des opérations se résume comme suit :

- Les données à jour sont enregistrées dans la mémoire permanente de la BDD,
- L'état précédent des données n'est plus disponible avec des requêtes SQL de lecture,
- Tous les utilisateurs peuvent voir les données résultant de la transaction.
- Les verrous sur les lignes affectées sont supprimés; les lignes sont donc accessibles aux autres utilisateurs et ils peuvent y apporter de nouvelles modifications.

LMD : COMMIT ET ROLLBACK

État des données après un ROLLBACK

L'exécution de la commande ROLLBACK annule les changements de données opérés par une transaction.

La séquence des opérations se résume comme suit :

- Les requêtes qui génèrent un changement des données sont annulées,
- L'état précédant des données est restauré,
- Les verrous sur les lignes affectées sont levés.

MEMOIRES DE LA BDD : DEFINITIONS

Lorsque l’on indique mémoire permanente et segments rollback, il s’agit d’espaces réservés par 1 instance Oracle.

D’un point de vue logique :

- La mémoire permanente est caractérisée par des blocs de données (data blocks),
- Les segments ROLLBACK permettent le stockage de segments d'images avant modification des données pour des annulations éventuelles (ROLLBACK).

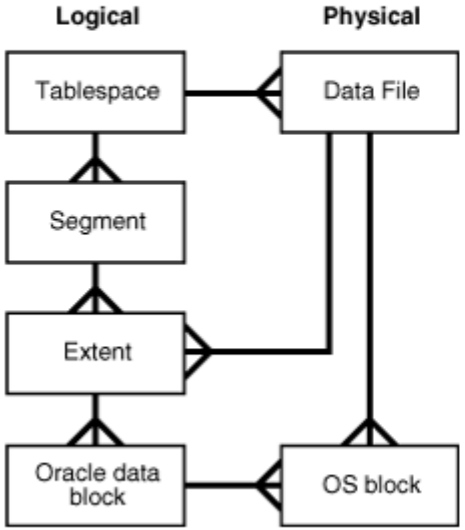
D’un point de vue physique :

Toutes les données sont enregistrées dans des data files (fichiers de données) sur 1 disque dur dans les clusters (ou blocks) du système d’exploitation.

MEMOIRES DE LA BDD : SCHEMA GENERAL

Figure 12-1 is an entity-relationship diagram for physical and logical storage. The crow's foot notation represents a one-to-many relationship.

Figure 12-1 Logical and Physical Storage



Description of "Figure 12-1 Logical and Physical Storage"

MEMOIRES DE LA BDD : TRANSACTION

Lorsque l'on exécute une transaction LMD, on a la séquence suivante :

1/

Avant que les requêtes d'écriture ne soient validées par un COMMIT, seul l'utilisateur qui est en train d'ajouter ou de modifier les données voit les modifications effectuées.

Les autres utilisateurs ne voient que des images du segment rollback.

=> Cela garantit que les autres utilisateurs visualisent des données consistantes et pas les données en cours d'ajout ou de mise à jour.

MEMOIRES DE LA BDD : TRANSACTION

2/

Si un COMMIT est exécuté sur des requêtes LMD :

Les changements faits sur la BDD deviennent visibles aux autres utilisateurs

L'espace occupé par les anciennes données dans le segment rollback est libéré pour réutilisation.

2bis/

Si un ordre ROLLBACK est exécuté, tous les changements sont annulés :

- Les valeurs originales (les valeurs antérieures) des données stockées dans les segments rollback sont réécrites dans la table.
- Tous les utilisateurs voient la BDD telle qu'elle existait avant que la transaction ne commence.