Table des matières

I. Introduction 2

II. La définition des données - LDD 4

III. Les fonctions 12

IV. Le contrôle des transactions 12

V. Approfondissement 12

Toute donnée est stockée dans une structure de base de données. On parle de BDD pour désigner le stockage des données et de SGBD pour désigner les éléments (système) qui sont mis à la disposition du développeur pour manipuler ces données.

Le langage SQL pour Structured Query Language – langage de requête structuré se décompose en plusieurs sous-ensembles :

- DDL- Data Définition Language, qui se composent principalement des ordres CREATE, ALTER, DROP. Il regroupe les ordres (commandes) utilisés pour créer, modifier et supprimer les structures (index, tables, vues, etc.) de la base de données.
- DML – Data Manipulation Language, qui se composent principalement des ordres SELECT, INSERT, DELETE, UPDATE. Il regroupe les ordres (commandes) utilisés pour manipuler les données dans la base.
- DCL – Data Control Language, qui se composent principalement des ordres GRANT et REVOKE. Il regroupe les ordres (commandes) utilisés pour gérer la sécurité des accès aux données.
- TCL – Transaction Control Language, qui se composent principalement des ordres COMMIT et ROLLBACK. Il regroupe les ordres (commandes) utilisés pour gérer la validation ou non de la mise à jour de la base données.

Il existe plusieurs normes SQL. La norme SQL2 ou SQL92 est la plus importante. C'est cette dernière qui est implémentée dans la majorité des SGBDR (système de gestion de base de données relationnelles). Ainsi, chaque fournisseur de SGBDR a implémenté sa façon le langage SQL et a ajouté ses propres extensions.

I. Introduction

Le modèle relationnel repose sur la notion d'ensemble. Un ensemble peut être représenté par une table (aussi appelée une relation). Cet ensemble a des attributs (colonnes) et des lignes (tuples). Une fois ces tables définies, il faut disposer d'un langage pour les manipuler. Il s'agit de l'algèbre relationnelle. Ainsi, l'algèbre relationnelle est mise en œuvre par le SQL et les systèmes de gestion de bases de données relationnelles implémentent le modèle relationnel. A l'aide des opérateurs, on peut interroger les relations existantes et créer de nouvelles relations.

On parle d'opérateurs ensemblistes : union, intersection, différence, produit cartésien, division et jointure.

Le modèle relationnel est caractérisé par trois concepts :

- La Relation, fondement du modèle relationnel. La relation permet de mettre en relation des domaines suivant certains critères. Le degré est le nombre d'attribut d'une relation.
- La Domaine, ensemble de valeurs caractérisées par un nom.
- Le Produit cartésien, représentant la jonction entre deux domaines.

L'algèbre relationnelle est la base théorique des opérations que l'on peut réaliser sur les bases de données relationnelles, et SQL (Structured Query Language) en est la concrétisation pratique. Le SQL est devenu le standard en ce qui concerne la gestion des données. Il permet la manipulation des tables et des colonnes. Son principe repose sur la création de nouvelles tables (tables résultantes) à partir des tables existantes. Ces nouvelles tables devenant des objets utilisables immédiatement. Les principales opérations de l'algèbre relationnelle entre deux tables table1 et table2 sont :

- L'Union entre deux relations de même structure (degré et domaines). La commande UNION renvoie toutes les lignes de table1 et table2, sans les doublons.
- L'Intersection entre deux relations de même structure (degré et domaines). La commande INTERSECT renvoie seulement les lignes qui apparaissent dans les deux tables
- La Différence entre deux relations de même structure (degré et domaines). La commande EXCEPT renvoie les lignes de table1 qui ne sont pas dans table2.
- La Division entre deux relations est possible à condition que la relation diviseur soit totalement incluse dans la relation dividende. Le quotient de la division correspond à l'information qui, présente dans le dividende, n'est pas présente dans le diviseur.
- La Restriction (Sélection) repose sur une condition. Elle produit, à partir d'une relation, une relation de même schéma n'ayant que les éléments de la relation initiale qui répondent à la condition. Elle s'utilise avec la commande WHERE condition.
- La Projection d'une relation sur un groupe d'attributs donne une relation résultante ayant comme schéma uniquement ces attributs, et comme éléments les n-uplets

distincts composés par les valeurs associées de ces attributs. Elle permet de supprimer les doublons. Extrait certaines colonnes d'une table, en supprimant les doublons. La commande DISTINCT colonne1, colonne2 FROM table1 renvoie les valeurs distinctes des colonnes spécifiées.

- La Produit cartésien combine toutes les lignes de deux relations pour produire toutes les combinaisons possibles. SELECT * FROM table1, table2 crée une combinaison de chaque ligne de table1 avec chaque ligne de table2.
- La Jointure entre deux relations est produite par la restriction sur le produit cartésien. La Jointure combine les lignes de deux relations basées sur une condition (généralement une égalité entre colonnes). SELECT * FROM table1 JOIN table2 ON table1.colonne1 = table2.colonne2 combine les lignes de table1 et table2 lorsqu'il y a correspondance des valeurs spécifiées
- La Calcul d'agrégats projection sur relation associée à un ou des calculs statistiques portant sur un attribut pour tous les éléments de la relation ou du regroupement lié à la projection afin de créer un ou plusieurs nouveaux attributs. COUNT, SUM, AVG, MAX, MIN.

II. La définition des données – LDD

Il existe trois grandes de familles de données : numérique, caractère (ou alphanumérique) et temporelle (dates et heures) :

- Numériques : les types numériques principaux de la nome SQL2 sont NUMERIC pour les données à virgule, INTERGER pour les entiers long et SMALLINT pour les entiers courts. La plupart, des SGBDR fusionnent le type NUMERIC et le type INTERGER en un seul type nommé NUMBER (ou DECIMAL). Notons cependant, qu'il existe d'autre types ajoutés par des SGBDR tels que TINYINT, SMALLINT, LONG, FLOAT.
- Caractères : le type alphanumériques classique se note CHAR pour les données de taille fixe et VARCHAR pour les chaînes de caractères de longueur variable. Dans MySQL, la taille maximum d'une chaîne CHAR et VARCHAR est normalement de 255 caractères. Si la saisie dépasse cette limite, MySQL transforme automatiquement le type en TEXT, MEDIUMTEXT ou LONGTEXT. Ces types sont de longueur fixe.

- Dates ou heures : Les types de format temporel sont principalement DATE, TIME et TIMESTAMP (ou DATETIME). La manipulation des dates et des heures est très différente d'un SGBDR.

1. La création de tables

CREATE est l'ordre de base en langage SQL pour créer un élément (TABLE, INDEX, VUE, SYNONYME). Cette commande est indépendante du stockage physique de la table. En effet, chaque SGBDR a sa propre syntaxe dans ce domaine. Dans la majorité des cas, ce sont les DBA (Database Administrator – administrateur de base de données) qui spécifient les normes de stockage et les options à appliquer sur les tables (en termes de performance et de sécurité de la base de données).

La clause COMMENT permet d'ajouter des commentaires sur les colonnes et tables directement dans la base de données au moment de sa création. Compte tenu du fait qu'une base de données à une durée de vie importante et va être utilisée par une population importante et hétéroclite, ajouter les commentaires participe à la maintenance et à l'évolution de la base.

Dans une table, il y'a souvent une colonne qui fait office d'identifiant unique (une sorte de compteur) et qui est la plupart du temps de type numérique. Pour laisser le SGBDR gérer l'incrémentation de cette colonne, on peut utiliser un objet de type SEQUENCE et l'associer à une colonne. Par ailleurs, certains SGBDR permettent de créer une séquence indépendamment d'une colonne de table. Oracle implémente l'objet séquence. Quant à MySQL, il ne l'implémente pas, il existe néanmoins la fonction AUTO_INCREMENT sur une colonne lors de la création de la table. Cependant, cette colonne doit être la PRIMARY KEY.

Dans une base de données, les tables sont rattachées à la personne (user) qui les a créées (souvent le DBA) qui donne le droit (ordre GRANT) aux autres intervenants d'utiliser ses tables. Par défaut, une table est préfixée par l'utilisateur qui l'a créée (ABOUBAKAR.TELEPHONE). Cependant on peut utiliser l'ordre SYNONYM pour simplifier un nom de table qui est normalisé, ou pour pointer sur une archive dont le nom n'a pas à être connu par les utilisateurs.

Une autre méthode pour créer une table consiste à dupliquer ou à s'inspirer de tables existantes. Pour récupérer uniquement la structure de la table d'origine, il faut ajouter une clause (condition) qui n'est jamais vérifiée.

Ligne de commandes
o MySQL : create table table2 select * from table1 o MySQL : create table table2 like table1 o ORACLE : create table table2 as select * from table1

2. La suppression de tables

La suppression de tables est une opération simple mais qu'il faut manier avec prudence, en effet, cette dernière est définitive. L'ordre DROP permet de supprimer définitivement une table (contenu, index, contraintes, commentaires). Cependant, cet ordre ne détruit pas les synonymes. On utilise souvent l'ordre DROP juste avant la création d'une table.

3. La modification de tables

L'ordre ALTER est utilisé pour réaliser plusieurs actions (supprimer ou ajouter une colonne d'une table d'une table, ajouter ou supprimer une contrainte ou ajouter une valeur par défaut à une colonne, et même modifier le type (CAST) d'une colonne).

L'ordre RENAME permet de renommer une table. Cette commande peut être utilisée dans le cas où la table doit être recréée tout en conservant la version actuelle en archive.

Ligne de commandes
o MySQL : alter table table_1 add/drop nom_2_col ou nom_2_contrain o MySQL : rename table to sav_table

4. Les vues

Les utilisateurs ou développeurs ont des besoins d'extractions spécifiques dans une base de données. Ces extractions sont matérialisées sous la forme de requêtes lancées manuellement ou incluses dans des programmes. Dans le cas, ou ces demandes sont répétitives ou communes à plusieurs utilisateurs, il est parfois nécessaire de créer une vue.

Une vue est une représentation logique de la base qui résulte d'une requête pour un besoin spécifique et répétitif. Contrairement à une table, une vue n'est pas stockée sur le disque. Les vues permettent de créer des tables « virtuelles » spécifiques pour un domaine ou pour une classe d'utilisateurs. Il est possible de créer un ensemble de vues par le type d'utilisateur ou métier de l'entreprise.

L'avantage majeure d'une vue est qu'elle est en permanence mise à jour. En effet, une vue est mise à jour automatiquement lorsque les tables qu'elles utilisent sont modifiées. Pour un utilisateur, une vue se comporte comme une table. Il peut réaliser ses requêtes sur la vue comme sur une table.

Ligne de commandes
○ create view <nom_vue> as select

En théorie, une vue n'est pas stockée sur le disque, elle est montée en mémoire lors de sa première utilisation. Cependant, certains SGBDR proposent de stocker des vues sur le disque, dans ce cas, elles se comportent comme des tables. L'accès à la vue se fera avec SELECT, comme pour une table classique. La suppression d'une vue se fait avec l'ordre DROP VIEW.

5. Les contraintes (intégrité des données) et les types de clés

Les contraintes (d'intégrité) en base de données relationnelle sont des règles définies pour garantir l'intégrité, la cohérence et la validité des données. Les contraintes qui imposent des restrictions sur les données stockées dans une table sont étroitement liées aux clés (primaires, étrangères, candidates, etc), car ces dernières définissent certaines règles.

En somme, une clé est un type particulier de contrainte. Elles imposent des règles particulières sur les données, toujours dans le but de garantir l'intégrité, l'unicité et la cohérence au sein d'une base de données.

5.1. Les principales contraintes

a. Contrainte NULL & NOT NULL

La contrainte NULL et NOT NULL, lors d'une insertion ou d'une modification, si l'on ne précise pas de valeur pour une colonne, celle-ci est vide et prend la valeur NULL. La contrainte NOT NULL impose comme contrainte qu'une colonne ne puisse pas contenir de valeur NULL. Par exemple, les colonnes d'une clé primaire sont automatiquement soumises à cette contrainte, car une clé primaire ne peut pas être NULL.

b. Contrainte DEFAULT

La contrainte DEFAULT, définit d'une valeur par défaut pour une colonne si aucune valeur n'est spécifiée lors de l'insertion d'un enregistrement. Elle peut permettre d'attribuer une valeur par défaut à cette colonne afin de ne pas avoir de valeur NULL dans la base. L'attribution d'une valeur par défaut s'effectue lors de la création de la table ou par un ALTER TABLE. Il est important de noter que cette valeur peut être la sortie d'une fonction comme la date du jour. Précisons que pour qu'une valeur par défaut soit prise en compte par le SGBDR, il faut qu'aucune valeur n'ai été affectée préalablement à la colonne via un ordre INSERT.

c. Contrainte UNIQUE

La contrainte UNIQUE impose que toutes les valeurs dans une colonne ou un groupe de colonnes soient uniques, c'est-à-dire qu'aucun doublon ne soit autorisé. Cette contrainte est utilisée pour garantir l'unicité des valeurs dans une colonne qui n'est pas une clé primaire. Cela est par exemple, une adresse email dans une table Utilisateurs. Il est à noté qu'une colonne déclarée en UNIQUE peut quand même contenir un ou plusieurs NULL, si l'on ne précise pas la clause NOT NULL au moment de sa création.

d. Contrainte CHECK

La contrainte CHECK impose que toutes les valeurs dans une colonne doivent satisfaire une condition spécifique. Cela permet de restreindre les valeurs possibles dans une colonne (par

exemple >0). ON peut contrôler la colonne avec des valeurs, mais également par l'appel à une fonction ou encore en spécifiant un SELECT spécifique. Il est a noté que la clause CHECK est implémentée dans MySQL, mais n'a aucun effet. Il faut passer par des contrôles par un programme ou utiliser TRIGGER.

5.2. Les clés

Les clés sont des éléments essentiels dans les bases de données relationnelles, car elles assurent l'intégrité des données et permettent de gérer les relations entre les tables. Il s'agit de contraintes spécifiques.

a. PRIMARY KEY

PRIMARY KEY : clé primaire (Primary Key), il s'agit de la clé principale d'une table. La contrainte de clé primaire impose que les valeurs dans les colonnes spécifiées doivent être uniques et non nulles. Le SGBDR va contrôler systématiquement à chaque insertion ou modification d'une ligne que la clé est unique dans la table. Dans le cas contraire, il rejette la demande de modification avec un message d'erreur de ce message d'erreur. Cette contrainte assure que chaque enregistrement dans la table est unique. La clé primaire est directement liée à cette contrainte. Lorsqu'une colonne ou ensemble de colonnes est désigné comme clé primaire, la contrainte PRIMARY KEY est automatiquement appliquée. Dans ce cas, la/les colonne(s) sera/seront NOT NULL et UNIQUE. Notons qu'il est conseillé de nommer la contrainte. Généralement, PK_<nom de table>, pour Primary Key est utilisé afin que développeur ou utilisateur à la lecture du nom contrainte sache qu'il s'agit d'une clé primaire et sur quelle table le problème a été rencontré.

b. SECONDARY KEY

SECONDARY KEY : clé secondaire, une clé secondaire est un index crée sur un ou plusieurs champs d'une table, en plus de la clé primaire. Contrairement à la clé primaire, la clé secondaire n'est pas nécessairement unique et peut contenir des valeurs NULL. Les clés secondaires ne servent pas à identifier de manière unique les enregistrements dans une table, mais plutôt pour accélérer les recherches basées sur des colonnes spécifiques. Sans un index secondaire, la base de données doit effectuer une recherche de manière séquentielle pour trouver ceux qui correspondent à la condition. Un index secondaire permet à la base de

données de localiser les données beaucoup plus rapidement. Les index secondaires peuvent considérablement améliorer les performances de requêtes complexes, notamment celles qui utilisent des clauses WHERE, ORDER BY ou JOIN sur les colonnes indexées. Cependant, la création et la maintenance des index secondaires ont un coût. Chaque fois qu'une opération INSERT, UPDATE ou DELETE est effectué sur une table, les index secondaires doivent être mis à jour, ce qui peut ralentir ces opérations. Trois types d'index existent : (1) index unique : empêche les valeurs dupliquées dans la colonne indexée, (2) index non unique, permet des valeurs dupliquées (type d'index secondaire le plus courant), (3) index composite qui englobe plusieurs colonnes, optimisant les requêtes qui filtrent ou trient sur ces données.

Les différences entre clé primaire et clé secondaire : une clé unique identifie de manière unique un enregistrement dans une table, une clé secondaire optimise l'accès aux données pour les opérations de recherche, tri ou filtrage.

c. FOREIGN KEY

FOREIGN KEY : clé étrangère, c'est-à-dire que l'on s'appuie sur une autre table pour indiquer comment contrôler les colonnes de notre table principale. Une clé étrangère est une contrainte qui lie une colonne d'une table à la clé primaire d'une autre table, garantissant que les relations entre les tables sont cohérentes. Ainsi, une fois que la table étrangère contienne une clé primaire pour que le SGBDR puisse faire le lien.

d. CANDIDATE KEY

CANDIDATE KEY : clé candidate, une clé candidate est un ensemble minimal d'attributs qui peut identifier de manière unique un enregistrement dans une table. Une table peut avoir plusieurs clés candidates (ID étudiant, email).

e. COMPOSITE KEY

COMPOSITE KEY : clé composite, une clé composite est une clé primaire composée de deux ou plusieurs colonnes qui ensemble identifient de manière unique un enregistrement dans une table. Les colonnes individuelles de la clé composite peuvent ne pas être uniques, mais leur combinaison l'est.

f. FOREIGN COMPOSITE KEY

FOREIGN COMPOSITE KEY : clé étrangère composite, similaire à la clé composite, une clé étrangère composite est une clé étrangère qui se compose de deux ou de plusieurs colonnes et fait référence à une clé composite dans une autre table.

g. SURROGATE & NATURAL KEYS

SURROGATE et NATURAL KEY : clé substitut et clé naturelle, une clé naturelle est un attribut ou un ensemble d'attributs existant naturellement dans les données et pouvant servir de clé primaire. Il s'agit généralement de donnée significative (comme un numéro de sécurité sociale, une adresse email, etc.). Une clé substitut est un identifiant artificiel ajouté à une table pour servir de clé primaire, souvent sous forme de numéro séquentiel ou UUID. Les clés substituts sont utilisées lorsque les clés naturelles sont complexes ou sujettes à un changement (Cela peut être un champ ID auto-incrémenté).

6. Les index

Le temps d'accès aux données est un paramètre crucial dans une base de données relationnelle. Ainsi, l'utilisation d'un index a pour but d'accélérer la recherche dans une base de données et s'appuie sur des fichiers physiques qui sont créés lors de la création de l'index. Sans le fichier d'index, l'ensemble de la table est parcouru séquentiellement jusqu'à trouver l'enregistrement demandé.

Les index ne font pas partie de la norme SQL, il s'agit d'une implémentation physique géré par le SGBDR. Il existe cinq méthodes de créations d'index : (1) hachage, (2) séquentiel, (3) bitmap, (4) arbre, (5) cluster. Les index les plus utilisés sont en bitmap et en arbre. Le choix de la méthode d'indexation est généralement de la responsabilité du DBA. L'index est le principal élément permettant d'améliorer les performances d'accès aux données. Cependant, la mise en place des index ralentit les traitements lors de la mise à jour, car le SGBDR doit recalculer les clés après chaque ajout/suppression/modification de lignes. En somme, SGBDR doit maintenir le fichier d'index à chaque mise à jour de la table.

Les index doivent être posées sur des colonnes avec des valeurs distinctives. L'index le plus performant sera sur une clé unique. Il est intéressant de donner un numéro aux index, ainsi on

peut connaître automatiquement le nombre d'index posés sur une table. Lorsque plusieurs colonnes.

Dans la pratique, il est conseillé de créer des index sur les colonnes qui sont déclarées en PRIMARY KEY, sur les colonnes en FOREIGN KEY, sur les colonnes les plus accédées, les colonnes qui servent de jointure entre les tables, les colonnes les plus discriminantes. Les index de type arbre convient aux tables volumineuses avec des clés uniques ou avec très peu de doublons (<5%). Les index de type bitmap sont à utiliser pour les tables volumineuses qui ont beaucoup de clés en commun avec des taux de mise à jour très faible. Dans une base très peu évolutive, il peut être intéressant de multiplier les index afin d'optimiser les temps de réponse. Cependant, il n'est pas nécessaire d'indexer les petites tables. En effet, l'analyse et la maintenance de l'index seront plus coûteuses que la lecture de la table complète. La suppression d'index se fait avec l'ordre DROP INDEX <nom index>.

Ligne de commandes
○ create [unique] index <nom_index> on <nom table> <nom colonne 1> [ASC/DESC] <nom colonne2>

III. Les fonctions

IV. Le contrôle des transactions

V. Approfondissement