Introduction aux Bases de Données

Le Langage SQL

Abdelali Saidi

saidi.a@ucd.ac.ma

Plan

1. Requêtes de définition de données
   - Création d'une base de données
   - Modification de la structure d'une table
   - Modification des champs d'une table
   - Suppression d'une table ou d'une base de données
2. Requêtes d'insertion de données
3. Requêtes de mise à jour des données
   - Modification des données
   - Suppression des données
4. Requêtes de sélection des données
   - Sélection avec critère de restriction
   - Jointures et unions
   - Fonctions de calculs et opérateurs d'agrégation
   - Les sous-requêtes

Introduction

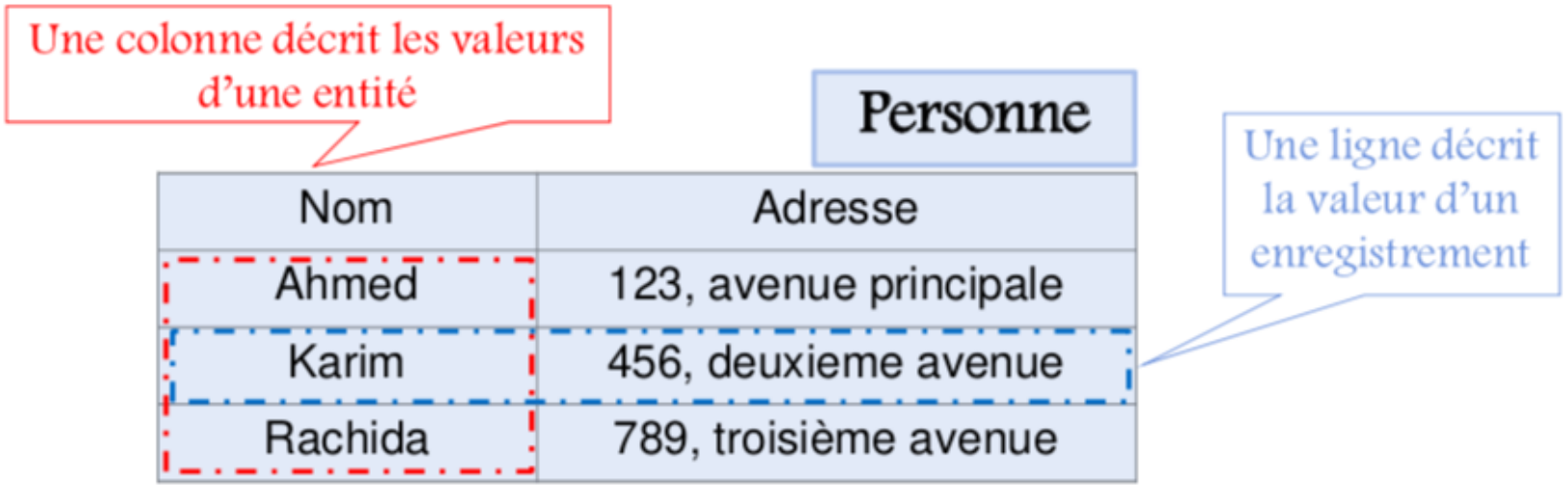
Présentation

- Structured Query Language
- Il permet de définir, administrer et manipuler les bases de données
- Il est exprimé sous forme de requêtes et de commandes
- Il est intégré aux SGBD[a]
  - Exemples de SGBD : Oracle server, MySQL, MariaDB, PostgreSQL, ...

[a] Système de gestion des bases de données

Introduction

Une table ?



Plan

Création d'une base de données

Création d'une base de données et ses tables

Pour être capable de stocker des données dans un SGBD, il faut suivre les étapes :

- Création d'une base de données
  - CREATE DATABASE NomBD
- Sélection de la base de données
  - USE NomBD
- Création des tables de la base de données
  - CREATE TABLE NomTable(...)

Remarque : Il est recommandé de mettre les mots clés en majuscule et le reste en minuscule

Création d'une base de données

Exemple

- CREATE DATABASE Test
- USE Test
- CREATE TABLE pays(id_pays,nom_pays,surface_pays,langue_pays,PRIMARY KEY(id_pays))
- CREATE TABLE auteur(id_aut int(5),nom_aut char(50),tel_aut char(10),PRIMARY KEY(id_aut), FOREIGN KEY(id_pays) REFERENCES pays(id_pays))

- DESC auteur : affiche les caractéristiques d'une table ainsi que ses enregistrements

Création d'une base de données

À la création d'une nouvelle table, il est possible de préciser quelques caractéristiques à propos de ses champs :

- PRIMARY KEY : pour indiquer la clé primaire
- FOREIGN KEY : pour indiquer la clé étrangère
- REFERENCES : pour indiquer la table d'où provienne une clé étrangère
- NOT NULL : pour empêcher une valeur nulle au niveau du champs concerné
- DEFAULT : pour indiquer la valeur par défaut
- AUTO_INCREMENT : insertion automatique d'une nouvelle valeur par incrément à chaque nouvelle enregistrement

Création d'une base de données

Exemple

Soient les deux relations suivantes :

- client (cli_num, cli_nom, cli_age)
- commande (cmd_num, #cli_num, cmd_date, cmd_statut)

Création des tables

- CREATE TABLE client (cli_num INTEGER NOT NULL, cli_nom VARCHAR(20) NOT NULL, cli_age INTEGER PRIMARY KEY (cli_num))
- CREATE TABLE commande (cmd_num INTEGER NOT NULL, cli_num INTEGER NOT NULL, cmd_date DATE NOT NULL, cmd_statut VARCHAR(20) PRIMARY KEY (cmd_num), FOREIGN KEY (cli_num) REFERENCES client)

Création d'une base de données

Type de données

Type	Taille (Octets)	Signification
Int	4	Valeur Entière
SmallInt	2	Valeur Entière
TinyInt	1	Valeur Entière
float	4/8	Valeur Décimale
Char (longueur)	Fixe (max 255)	Chaîne de caractères
VarChar (longueur)	Var (max 255)	Chaîne de caractères
Text	Var (max $2^{31}-1$)	Chaîne de caractères
Image	Var (max $2^{31}-1$)	Chaîne binaire
Bit	1	Valeur Binaire
Binary	Fixe (max 255)	Chaîne binaire
DateTime	2×4 octets	Nb Jours depuis 1/1/1900 + Heure

Modification de la structure d'une table

Après la création d'une nouvelle base de données et ses tables, il n'est pas rare de vouloir modifier sa structure (modifier une colonne, supprimer une colonne, supprimer une table ...)

- Syntaxe : ALTER TABLE nomTable ADD|DROP|CHANGE|MODIFY
  - Ajouter une nouvelle colonne : ADD nomCol typeDonnée
  - Ajouter une contrainte NOT NULL : ADD CHECK nomCol NOT NULL
  - Ajouter une clé étrangère : ADD CONSTRAINT FOREIGN KEY (nomCol) REFERENCES nomTable(nomCol)
  - Supprimer une colonne : DROP nomCol
  - Changer le nom d'une colonne : CHANGE ancienNom nouveauNom typeDonnée
  - Modifier le type de donnée : MODIFY nomCol nouveauType

Modification des champs d'une table

Si on veut ajouter un nouveau champs

Si on veut supprimer un champs

Modification des champs d'une table

Si on veut ajouter un nouveau champs

- ALTER TABLE NomTable ADD NomChamps type
- Exemple : ALTER TABLE auteur ADD dateNaissance date

Si on veut supprimer un champs

- ALTER TABLE NomTable DROP NomChamps
- Exemple : ALTER TABLE auteur DROP tel_aut

Suppression d'une table ou d'une base de données

- Supprimer une table : DROP TABLE nomTable
- Supprimer une base de données : DROP DATABASE nomBD

Plan

Insertion des données

La commande INSERT

On peut utiliser cette commande pour insérer un nouvel enregistrement dans une table

- INSERT INTO champ1,champ2,... NOMTABLE VALUES(...)
  - Il faut prendre en compte l'ordre des champs de la table
  - La lise des champs d'est pas obligatoire si on compte renseigner toutes les valeurs

Insertion des données

Exemple

Insérez les deux enregistrements suivants :

- (100,Hugo,12345678)
- (120,Maalouf,847365252)

Insertion des données

Exemple

Insérez les deux enregistrements suivants :

- (100,Hugo,12345678)
- (120,Maalouf,847365252)
- INSERT INTO auteur VALUES(100, "Hugo" ,12345678)
- INSERT INTO auteur VALUES(120, "Maalouf" ,847365252)

Plan

Modification des données

La commande UPDATE

On peut utiliser cette commande pour modifier les valeurs de champs d'un enregistrement donné

- UPDATE NomTable SET operationMàJ WHERE condition

Modification des données

Exemple

Modifiez le numéro de tél de Maalouf. Le nouveau numéro est : 020303039

Modifiez l'identifiant de Maalouf

Modification des données

Exemple

Modifiez le numéro de tél de Maalouf. Le nouveau numéro est : 020303039

- UPDATE auteur SET tel_aut=020303039 WHERE nom="Maalouf"

Modifiez l'identifiant de Maalouf

- UPDATE auteur SET id_aut=122 WHERE nom="Maalouf"

Suppression des données

La commande DELETE

On peut utiliser cette commande pour supprimer un enregistrement donné

- DELETE FROM NomTable WHERE condition : suppression de quelques enregistrement
- TRUNCATE table NomTable : suppression de tous les enregistrement

Remarque : il est possible de supprimer tous les enregistrements avec DELETE FROM sans à utiliser la clause WHERE. Toutefois, TRUNCATE est plus rapide, n'accepte pas l'utilisation de cette clause et ne peut pas être annulé par un ROLLBACK.

Suppression des données

Exemple

Supprimez de la table auteur l'enregistrement dont le nom est Hugo

Et si on veut supprimer toute la table ?

Suppression des données

Exemple

Supprimez de la table auteur l'enregistrement dont le nom est Hugo

- DELETE FROM auteur WHERE nom_aut='Hugo"

Et si on veut supprimer toute la table ?

- DROP TABLE auteur

Suppression des données

Clé étrangère

À la mise à jour ou la suppression d'un enregistrement possédant une clé étrangère, vis-à-vis des enregistrements qui sont liés, il est possible de choisir l'un des comportements suivants :

- CASCADE : suppression ou modification en cascade de tous les enregistrements concernés
- SET NULL : mettre la valeur NULL dans la case de la clé étrangère
- NO ACTION ou RESTRICT : c'est l'option par défaut, elle retourne une erreur en cas de modification ou suppression d'une valeur d'une clé étrangère

Plan

Sélection des données

La commande SELECT

On peut utiliser cette commande pour sélectionner et afficher les champs des enregistrements désirés

- SELECT champs1,champs2,champs... FROM NomTable

Si on veut afficher tous les champs, on peut utiliser le caractère *

- SELECT * FROM NomTable

Exemple

Affichez l'identifiant et le nom de tous les auteurs

Affichez toute la table auteur

Sélection des données

La commande SELECT

On peut utiliser cette commande pour sélectionner et afficher les champs des enregistrements désirés

- SELECT champs1,champs2,champs... FROM NomTable

Si on veut afficher tous les champs, on peut utiliser le caractère *

- SELECT * FROM NomTable

Exemple

Affichez l'identifiant et le nom de tous les auteurs

- SELECT id_aut,nom_aut FROM auteur

Affichez toute la table auteur

- SELECT * FROM auteur

Sélection avec critère de restriction

La clause WHERE

Cette cluse permet de préciser les enregistrements visés par une requête de sélection en indiquant la valeur d'une colonne ou plusieurs. La condition qui suit cette clause est définie par les opérateurs suivants :

- $=, <, >, <>$
- BETWEEN, LIKE
- IS NULL, IS NOT NULL, IN, NOT IN

Sélection avec critère de restriction

Exemple

Affichez le numéro de téléphone de l'auteur Maalouf

Affichez toutes les informations de l'auteur qui a comme identifiant la valeur 100

Affichez les noms de tous les utilisateurs qui ont 0660 au début de leur numéro de téléphone

Affichez les noms de tous les utilisateurs qui ont des identifiants entre 30 et 40

Affichez les numéros de téléphone des auteurs Einstein, Maalouf et Hugo

Sélection avec critère de restriction

Exemple

Affichez le numéro de téléphone de l'auteur Maalouf

- SELECT tel_aut FROM auteur WHERE nom_aut= "Maalouf"

Affichez toutes les informations de l'auteur qui a comme identifiant la valeur 100

- SELECT * FROM auteur WHERE id_aut¡100

Affichez les noms de tous les utilisateurs qui ont 0660 au début de leur numéro de téléphone

- SELECT nom_aut FROM auteur WHERE tel_aut LIKE 0660%

Affichez les noms de tous les utilisateurs qui ont des identifiants entre 30 et 40

- SELECT nom_aut FROM auteur WHERE id_aut BETWEEN 30 AND 40

Affichez les numéros de téléphone des auteurs Einstein, Maalouf et Hugo

- SELECT tel_aut FROM auteur WHERE nom_aut IN ("Einstein","Maalouf","Hugo")

Sélection avec jointure

Il est possible qu'on voudrait sélectionner des données appartenant à plusieurs tables ayant quelques colonnes en commun. Pour cela, on peut utiliser des jointures.

- Syntaxe : SELECT colonne1,colonne2,... FROM table1,table2,... WHERE table1.colCommune=table2.colCommune [AND autresConditions]

Exemple

- Affichez les noms des auteurs marocains

Sélection avec jointure

Il est possible qu'on voudrait sélectionner des données appartenant à plusieurs tables ayant quelques colonnes en commun. Pour cela, on peut utiliser des jointures.

- Syntaxe : SELECT colonne1,colonne2,... FROM table1,table2,... WHERE table1.colCommune=table2.colCommune [AND autresConditions]

Exemple

- Affichez les noms des auteurs marocains
- SELECT nom_aut FROM auteur,pays WHERE auteur.id_pays=pays.id_pays and nom="Maroc"

Sélection avec union

Le mot clé UNION permet d'unir les résultats de plusieurs requêtes de sélection.

- Syntaxe : SELECT colonne1,colonne2,... FROM table1 WHERE conditions UNION SELECT colonne1,colonne2,... FROM table2 WHERE conditions

Exemple

Sélection avec union

Le mot clé UNION permet d'unir les résultats de plusieurs requêtes de sélection.

- Syntaxe : SELECT colonne1,colonne2,... FROM table1 WHERE conditions UNION SELECT colonne1,colonne2,... FROM table2 WHERE conditions

Exemple

- SELECT nom_aut FROM auteur WHERE id_pays = 5 UNION SELECT nom_pays FROM pays WHERE id_pays = 5

Fonctions de calculs et opérateurs d'agrégation

- Fonction de calcul : COUNT, MAX, MIN, AVG, SUM
- Opérations d'Agrégation : GROUP BY, HAVING, ORDER BY

Fonctions de calculs et opérateurs d'agrégation

La fonction COUNT()

Cette fonction permet de compter le nombre d'enregistrements dnas une table

- SELECT COUNT(*) FROM auteur
- SELECT COUNT(*) FROM auteur WHERE ville_aut = 'El Jadida'
- SELECT COUNT(DISTINCT nom_pays) FROM pays

Fonctions de calculs et opérateurs d'agrégation

La fonction ORDER BY

L'opérateur ORDER BY permet d'afficher le résultat d'une sélection ordonnée.
Exemple :

- SELECT id_aut,nom_aut FROM auteur ORDER BY nom_aut
  - Le critère par défaut de l'ordre est la clé primaire
  - L'ordre est par défaut accroissant
  - DESC pour indiquer un ordre décroissant
  - Il est possible de désigner plusieurs critères d'ordre

Sélection avec suppression des doublons

L'opérateur DISTINCT permet d'afficher le résultat d'une sélection sans doublons.
Exemple :

- SELECT DISTINCT id_aut,nom_aut FROM auteur

Fonctions de calculs et opérateurs d'agrégation

La fonction GROUP BY

Cette fonction permet de grouper plusieurs résultats et utiliser une fonction de totaux sur un groupe de résultat.

- SELECT * FROM auteur GROUP BY ville_aut

Fonctions de calculs et opérateurs d'agrégation

La fonction HAVING

La condition HAVING en SQL est presque similaire à WHERE à la seule différence que HAVING permet de filtrer en utilisant des fonctions telles que SUM(), COUNT(), AVG(), MIN() ou MAX().

- Affichez les noms et prénoms des auteurs pour lesquels leur nom est partagé entre plus que 3 auteurs

Fonctions de calculs et opérateurs d'agrégation

La fonction HAVING

La condition HAVING en SQL est presque similaire à WHERE à la seule différence que HAVING permet de filtrer en utilisant des fonctions telles que SUM(), COUNT(), AVG(), MIN() ou MAX().

- Affichez les noms et prénoms des auteurs pour lesquels leur nom est partagé entre plus que 3 auteurs
- SELECT * FROM auteur GROUP BY nom_aut HAVING COUNT(nom_aut) > 3

Les sous-requêtes

Il est possible d'utiliser le résultat d'une sélection dans la clause WHERE d'une autre sélection.

- Syntaxe : SELECT col1,col2,... FROM table1,table2,... WHERE col OPERATEUR (une autre sélection)

Exemple

- Affichez toutes les informations des auteurs ayant l'identifiant compris entre l'identifiant de Einstein et l'identifiant d'Hugo

Les sous-requêtes

Il est possible d'utiliser le résultat d'une sélection dans la clause WHERE d'une autre sélection.

- Syntaxe : SELECT col1,col2,... FROM table1,table2,... WHERE col OPERATEUR (une autre sélection)

Exemple

- Affichez toutes les informations des auteurs ayant l'identifiant compris entre l'identifiant de Einstein et l'identifiant d'Hugo
- SELECT * FROM auteur WHERE id_aut BETWEEN (SELECT id_aut FROM auteur WHERE nom_aut = "Einstein") AND (SELECT id_aut FROM auteur WHERE nom_aut = "Hugo")