Bases de données

Il existe de nombreux systèmes de gestion de bases de données (SGBD), ils s'appuient sur le langage SQL.
Exemples de SGBD libres :
- SQLite (SQLite Manager est une extension du moteur de recherche Firefox). simple, licence domaine public, fichiers portant l'extension .sqlite (non compatibles avec d'autres SGBD).
- MySQL : licence GPL. On peut installer chez soi un serveur local et travailler en ligne de commande. Chaque instruction est suivie d'un point-virgule.
- PostgreSQL.

1 Tables

Une base de données est constituées de tables. Chaque table est un tableau 2D déterminé par ses attributs (noms des colonnes) et constituée de lignes (enregistrements).
Une clé primaire est un attribut (ou un ensemble d'attributs) permettant d'identifier de manière unique chaque ligne.
Lorsqu'il y a plusieurs tables, une clé étrangère référence une clé primaire d'une autre table.

Un modèle relationnel est une façon de modéliser les relations existantes entre plusieurs informations et de les ordonner. Une base de données permet d'implémenter un modèle relationnel.
Les associations entre tables sont assurées par les clés étrangères. On les représente dans un modèle relationnel par des flèches entre différentes tables.

2 Construction et modification

Ce chapitre est hors programme.

- CREATE DATABASE <nom_bdd> : crée la base de données.
- USE <nom_bdd> : charge la base de données.
- SHOW DATABASES : affiche les bases de données disponibles.
- CREATE TABLE <nom_table> (nom_att1 type_att1, nom_att2 type_att2,....) : crée une table en donnant la liste de ses attributs avec leur type.
  Les types de données possibles sont assez nombreux, de taille fixe ou variable, parmi lesquels INTEGER, REAL, BOOLEAN, VARCHAR (variable), FLOAT (variable) .
- SHOW TABLES : affiche la liste des tables de la base.
- DESCRIBE <nom_table> : affiche la structure de la table.
- INSERT INTO <nom_table> VALUES (val_att1,val_att2,...) : insère un enregistrement dans la table.
  L'instance vide est notée NULL.
- UPDATE <nom_table> SET <nom_att = new_val> WHERE <cond> : modifie la valeur d'un attribut selon une condition.
- DELETE FROM <nom_table> WHERE <cond> : supprime les lignes vérifiant une condition.
- ALTER TABLE <nom_table> [ADD <nom_att1 type_att1> AFTER <nom_att2>] [DROP COLUMN nom_att] [MODIFY nom_att new_type_att] : modifie la structure d'une table, en ajoutant ou supprimant ou modifiant le type d'un attribut.
- DROP <nom_table> : supprime une table.

Tout ce qui concerne cette partie est rarement demandé aux concours.

3 Requêtes

La syntaxe d'une déclaration de requête est la suivante : SELECT <liste d'expressions>
FROM <liste de tables>
WHERE <conditions>
GROUP BY <liste d'attributs>
HAVING <conditions>
ORDER BY <liste d'attributs>

- La clause SELECT spécifie le schéma de sortie.

  SELECT est suivi d'une liste d'expressions. On peut avoir des doublons. Pour les éliminer, il faut utiliser DISTINCT dans la clause SELECT.
  Exemple : SELECT DISTINCT $A_1, A_2, ..., A_n$ FROM $R_1, R_2, ..., R_m$ WHERE <cond$(A_1, ..., A_n, ...A_{n+k})$ >

- La clause FROM précise les tables impliquées et leurs liens (produit cartésien et jointures).

  SELECT * FROM table1,table2 : effectue le produit cartésien des deux tables.

  SELECT * FROM table1 JOIN table2 ON x = y : effectue un produit cartésien des 2 tables puis une sélection lorsque les attributs x et y ont la même valeur. On peut utiliser des alias à l'aide du mot-clé AS. On peut effectuer plusieurs jointures.

- La clause WHERE fixe les conditions que doivent remplir les n-uplets solutions par l'intermédiaire de différents opérateurs.

  Opérateurs de comparaisons : <,<=,>,>=, LIKE, BETWEEN.

  BETWEEN : plage de valeurs. Syntaxe : BETWEEN <val1> AND <val2>

  LIKE : comparaison partielle de chaînes de caractères.

  % : remplace n'importe quelle chaîne.
  - : remplace juste un caractère.
  LIKE 'ch %' : filtre ce qui commence par la chaîne 'ch'.
  LIKE 'ch % ch' : filtre ce qui commence par la chaîne 'ch'.

  <att> IS NULL : pas de valeur pour cet attribut.

  IN : liste de valeurs possibles. IN(val1,val2,...)

  Opérateurs sur les nombres : ABS, MOD, LN, EXP, POWER, FLOOR.

  Opérateurs sur les booléens : IS TRUE, IS FALSE.

  LIMIT : limite le nombre de réponses.
  LIMIT a,n : affiche n lignes à partir de la ligne a.

- La clause GROUP BY indique comment regrouper des n-uplets en précisant les attributs de groupement. Les fonctions d'agrégation sont ensuite appliquées sur chaque groupe.

  Les principales fonctions d'agrégation sont COUNT, SUM, AVG, MIN, MAX.

  COUNT : COUNT(*) compte le nombre de lignes.
  COUNT(att) : compte le nombre de lignes remplies pour cet attribut.
  SUM (att) : somme l'attribut att sur les colonnes sélectionnées.
  AVG (att) : calcule la moyenne de att.
  MIN (att) : affiche la plus petite valeur de att (idem avec MAX (att)).

- La clause HAVING permet de spécifier des conditions sur les groupes.

  WHERE agit sur les lignes avant le groupage, alors que HAVING effectue une sélection sur la table après calcul de la fonction agrégative.

- La clause ORDER BY permet de trier les résultats sélectionnés selon un critère (par exemple un attribut). Le suffixe DESC trie dans l'ordre décroissant.

- Opérations ensemblistes : UNION, INTERSECT, EXCEPT.
  L'intersection de deux tables suppose que les attributs des deux tables soient de même format.
  L'union de deux tables suppose que les attributs des deux tables soient en même nombre et de même type.

  Exemple de syntaxe : SELECT * FROM table1 INTERSECT SELECT * FROM table2