Introduction

Le but de ce projet est de réaliser un programme capable de simuler les principales fonctionnalités d’un système de gestion de base de données. Ainsi, le développement de ce programme nous permettra de mieux appréhender les problématiques auquel répond un vrai SGBD, et nous prendrons conscience de son rôle et de son importance.

Structure du projet

La réalisation de ce projet a débuté par l’analyse du sujet et de ses exigences. De cette analyse, nous en avons déduis que le programme demandé était divisé en trois grand fonctionnalités :

* La gestion des métadonnées de la base de données
* La gestion des n-uplets de la base de données
* L’interface utilisateur.

Dans cette partie, nous allons expliquer comme nous avons traités chacune de ces trois fonctionnalités, tant d’un point de vue fonctionnel (rôle de la fonctionnalité) que d’un point de vue technique (structure du code, comment a été pensé le développement). Cependant, nous n’aborderons pas les algorithmes de ces fonctionnalités, qui seront détaillés dans la prochaine partie.

1. Gestion des métadonnées

Les métadonnées ont pour rôle de décrire le schéma de la base de données. Le SGBD a ainsi connaissance de ses caractéristiques, à savoir le nom des relations de la base, quels sont leurs attributs, le domaine des attributs, etc… elles sont donc essentielles au fonctionnement du SGBD, car c’est ce schéma qui structure nos données.

Afin de mieux exposer ce que représente le schema, nous allons maintenant présenter la facon dont nous avons inclus cette notion dans notre programme.

Le schema contient les informations de nos relations, le nombre de relations, ainsi que le nom de la base de données.

typedef struct \_schema\_BD {

Meta\_relation \*\*relations;

int nb\_relations;

char nom\_BD[32];

} Schema\_BD;

Ensuite, chaque relation possède un id (son nom), un nombre d’attributs, ainsi qu’un tableau contenant ses attributs.

typedef struct \_meta\_relation {

char id[3];

int nb\_attr;

Meta\_attribut \*\*attributs;

} Meta\_relation;

Chaque attribut possède un identifiant (limité à 2 caractères + 1 caractère de fin de chaine), un domaine, ainsi qu’un rang (ordre de définition de l’attribut vis à vis de la relation)

typedef struct \_meta\_attribut {

char id[3]; //Identifiant (nom) de l'attribut

Meta\_domaine \*domaine;

int rang;

} Meta\_attribut;

Enfin, chaque domaine possède un type (0 si int, 1 si varchar), et un entier « taille » qui désigne le nombre d’octets nécessaire pour stocker l’attribut.

typedef struct \_meta\_domaine {

// Type \*type; //Type (int ou varchar)

int type;

int taille; // Taille en octets (si int alors 4 octets)

} Meta\_domaine;

Ainsi, le point d’entrée de tous ces élément est le schema : il possède toutes les informations concernant les relations, qui elles-même contient toutes les informations sur les attributs, qui eux détiens les informations sur leur domaines. L’ensemble de ces informations sont nécessaire pour décrire l’ensemble de la base de données.