# LEWIS Arnaud 21 Mai 2012 MLUDEK Denis STEYER Valentin TERRIER Hélène

# PROJET ADA :

# RAPPORT INTERMEDIAIRE

## U:\Documents\universite-de-lorraine.jpgSujet 1 : Réalisation d’un mini SGBD

# Analyse de l'application à réaliser:

Nous devons réaliser une application qui permet de créer et de gérer une base de données. Pour cela il va falloir commencer par définir les types permettant d'établir une structure logique. Pour cela on cherche à traduire en langage ADA le fonctionnement d'une base de données afin de réaliser un SGBD.

Il faut permettre à l'utilisateur de créer, modifier ou supprimer de nouvelles tables constituées d'attributs qui peuvent être de différents types (integer, string, date...) gérés par le biais d'un paquetage générique.

A la création d'une base, il y a création d'un fichier à accès séquentiel qui portera le nom de la base préfixé par "F\_" et suffixé par ".DAT". L'ensemble des données de la base sera stocké dans ce fichier. Pour gérer la base on aura besoin de créer ces fonctions:

- créer base

- modifier base

- supprimer base

- sauvegarder base

Concernant la gestion des tables, il y aura des structures imbriquées les une dans les autres qui contiennent des tableaux.

Une table est caractérisée par son nom et les différents attributs qui la composent.

Afin de gérer les bases de données de façon plus simple, chaque table et chaque attribut sera défini par un numéro (num\_tab et num\_attribut). Ainsi, quand on voudra modifier une table ou son contenu il suffira de connaitre son numéro ainsi que le numéro des attributs à modifier et de demander à l'utilisateur de rentrer les nouvelles valeurs.

De la même manière lors de la suppression d'une table ou d'un attribut il faudra passer leur numéro à -1. Une fois qu'une table est supprimée il faut que les suivantes prennent le numéro de la table précédente. Il en est de même lors de la suppression d'un attribut.

Il faudra donc trier les tables et les attributs sans oublier de créer une fonction qui déterminera l'indice de fin de tableau que ce soit pour les attributs ou pour les tables. Il en est de même lors de la suppression d'un attribut.

# Structure de données :

Afin de faciliter la compréhension des structures de données utilisées dans notre programme, voici un extrait du code source et un schéma.

*====================================================================*

*========================== Déclarations ===============================*

*====================================================================*

*type T\_Donnees is record*

*Carac\_Attribut:T\_Cle:=Neutre;*

*Valeur\_Attribut:T\_Element;*

*end record;*

*type T\_Contenu is array (Natural range 0..10) of T\_Donnees;*

*type T\_Attribut is record*

*Nom\_Attribut:T\_String;*

*Num\_Attribut:integer;*

*Type\_Attribut:T\_Element;*

*Contenu\_Attribut:T\_Contenu;*

*end record;*

*type T\_Tab\_Attribut is array (Natural range 0..10) of T\_Attribut;*

*type T\_Table is record*

*Nom\_Table:T\_String;*

*Num\_Table:integer;*

*Contenu\_Table:T\_Tab\_Attribut;*

*end record;*

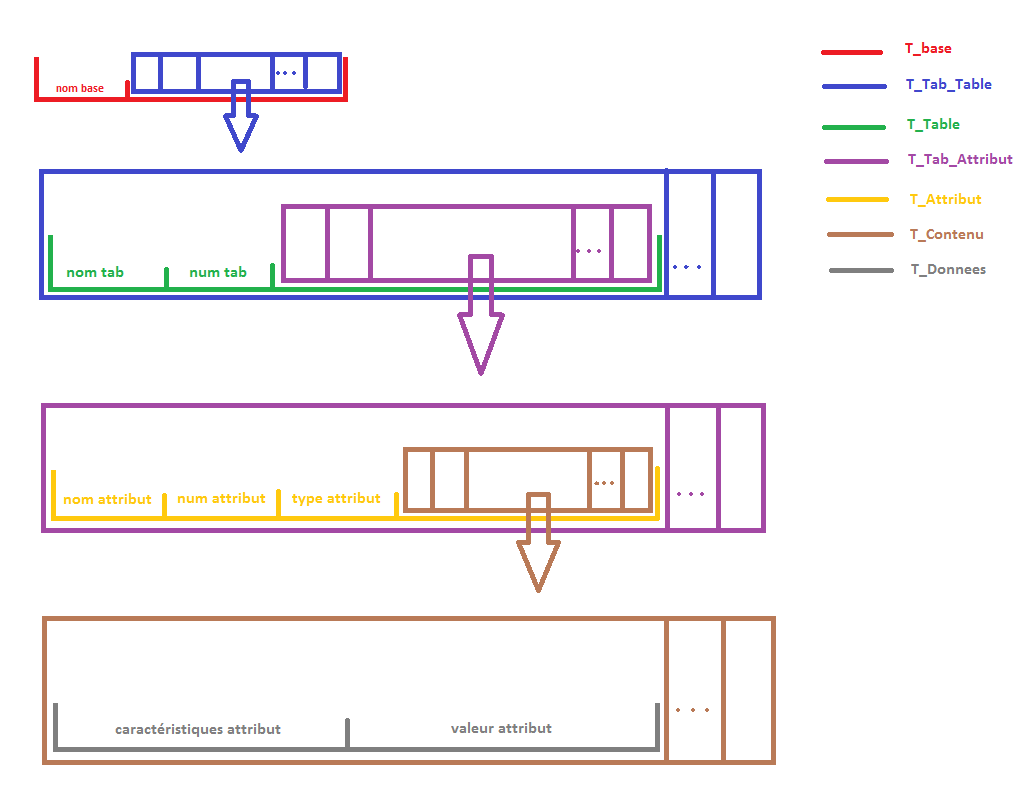
*type T\_Tab\_Table is array (Natural range 0..10) of T\_Table;*

*type T\_Base is record*

*Nom\_Base:T\_String;*

*Contenu\_Base:T\_Tab\_Table;*

*end record;*

Schéma :

# Choix de programmation :

Nous avons choisi de programmer en mode console, par la suite nous essayerons d'intégrer une interface graphique.

Afin de structurer le projet, nous utiliserons un paquetage générique car le type d'un attribut dépend du choix de l'utilisateur.

Ci-dessous quelques entêtes de fonctions et/ou de procédures usuelles. C’est une première réflexion, elles ne sont pas définitives, d’autres s’y ajouteront par la suite.

*-- fonctions sur la base*

*procedure Create\_Base(B:in T\_Base; Nom\_Fic:in out String; Taille:in Integer);  
procedure Open\_Base(B:out T\_Base; Nom\_Fic:in string);  
procedure Save\_Base(B:in T\_Base; Nom\_Fic:in string);  
procedure Close\_sgbd;*

*-- fonctions de gestion de tables  
  
function Indice\_Fin\_Table(B:in T\_Base) return Integer;  
function Tri\_Table(B:in T\_Base) return T\_Base;  
function Create\_Table(B:in T\_Base; Nom\_Table:in string) return T\_base;  
function Delete\_Table(B:in T\_base; T:in T\_Table) return T\_Base;  
function Modif\_Table(B:in T\_base; T:in T\_table) return T\_base  
function Remplir\_Table(B:in T\_Base; Table:in T\_Table; T:in T\_Contenu) return T\_Base ;  
procedure Affiche\_Contenu\_Table(B:in T\_Base; Table:in T\_Table);  
procedure Affiche\_Carac\_Table(B:in T\_Base; Table:in T\_table);  
procedure Affiche\_Liste\_Tables(B:in T\_Base) ;  
  
 -- fonctions de gestion d’attributs*

*function Add\_Attribut(B:in T\_Base; Table:in T\_Table; Nom\_Attribut:in T\_String; Type\_Attribut:in T\_Element) return T\_Base;  
function Delete\_Attribut(B:in T\_Base; Table:in T\_Table; Attribut:in T\_Attribut) return T\_Base;  
function Modif\_Attribut(B:in T\_Base; Table:in T\_Table; Attribut:in T\_Attribut) return T\_Base;*

**Répartition des tâches:**

Fonctions sur la base : Denis MLUDEK  
Fonctions de gestion des tables : Hélène TERRIER & Valentin Steyer  
Fonctions de gestion d’attributs : Arnaud LEWIS

L'interface graphique et le programme principal feront l’objet d’une réflexion et d’un travail commun.