

PROJET DE SGBD DIC 1 INFORMATIQUE

Travail realisé par :

* Cherif Younouss NDIAYE
* Serigne Cheikh Mbacke THIAW

.

INTRODUCTION

Afin de faciliter l’analyse de flux de donnees au format JSON ou XML, il nous a été demandee de developper une solution d’ingenierie inversee. La dite solution devra permettre de generer un fichier SVG permettant de visualiser un diagramme entite-association refletant la structuration des donnees qu’elle aura en entree

Nous avons utilisee le langage de programmation C afin de pouvoir realiser la solution.

Dans la suite, on va d’abord presenter la structuration du projet et enfin les fonctions qui nous ont permis de realiser les differents traitements relatifs aux differents flux de donnees

STRUCTURATION DU PROJET

Le projet comprend essentiellement trois modules :

* Module JSON : Dans ce module sont effectuees tous les traitements relatifs au flux de donnees de type JSON
* Module XML : Dans ce module sont effectuees tous les traitements relatifs au flux de donnees de type XML
* Module SVG : Ce module traite de la generation du fichier SVG qui doit comporter le diagramme d’entite-association

Dans la suite, nous allons voir les differentes fonctions qui caracterisent ces modules.

DESCRIPTION DES FONCTIONS OU PROCEDURES UTILISEES

1. Module JSON
   1. Validation de donnees  
      Pour la validation de donnees au format JSON, la librairie JSON\_CHECKER a été utilisée.  
      Les fonctions suivantes ont été notamment utilisées :
      * int est\_valide (char\* filename) : Cette fonction prend en argument le nom du fichier à valider et retourne 0 si la validation reussi. Sinon, elle arrete le programme par le biais de la fonction exit.
      * int JSON\_checker\_char(JSON\_checker jc, int next\_char) : Il s’agit ici d’une fonction de la librairie JSON\_CHECKER qui permet de faire la validation d’une structure JSON caractere par caractere
      * int JSON\_checker\_done(JSON\_checker jc) : Il s’agit d’une fonction de la librairie JSON\_CHECKER qu’on appelle apres que tous les caracteres ont été. Elle retourne alors VRAI si la verification a reussi
   2. Extraction de donnees  
      Pour l’extraction des donnees au format JSON, la librairie cJSON a été utilisee  
      Les fonctions suivantes peuvent y etre notees :
      * char\* chaine\_json(char\* filename) : Cette fonction permet de parcourir un fichier et de retourner son contenu sous format d’une chaine de caracteres
      * void get\_nombre\_entites\_associations(cJSON\* json, int\* nb\_entites, int\* nb\_associations) : Cette procedure permet de recuperer le nombre d’entite et le nombre d’associations
      * char\*\* get\_attributs\_entite(cJSON\* json, char\*\* attributs, int\* n\_attributs) : Cette fonction permet de recuperer les attributs d’une entite
      * char\*\* get\_attributs\_association(cJSON\* json, char\*\* attributs, int\* n\_attributs) : Cette fonction permet de recuperer les attributs d’une association
      * void test(cJSON\* json, s\_entite\*\* tab\_entites, s\_association\*\* tab\_associations) : Cette fonction permet de parcourir le premier niveau de l’arborescence du fichier JSON
      * void test2(cJSON\* json, s\_entite\*\* tab\_entites, s\_association\*\* tab\_associations) : Cette fonction permet de realiser le parcours recursif de l’arborescence du fichier json
      * char\*\* get\_associations(cJSON\* json, cJSON\* parent, char\*\* associations, int\* n\_associations) : Cette fonction permet de recuperer les associations auxquelles participent l’entite
      * void afficher\_entite(s\_entite\* entite) : Fonction permettant d’afficher une entite pris en argument
      * void afficher\_association(s\_association\* association) :Fonction permettant d’afficher une association
      * s\_entite\* creer\_entite(s\_entite\* e, char\* nom, char\*\* attributs, char\*\* associations, int n\_attributs, int n\_associations) : Fonction qui cree et retourne une entite
      * s\_association\* creer\_association(s\_association\* a, char\* nom, char\*\* attributs, int n\_attributs) : Fonction qui cree et retourne une association
      * void test2\_bis(cJSON\* json, s\_entite\*\* tab\_entites, s\_association\*\* tab\_associations) : Fonction pour faire l’extraction de nœuds de meme niveaau
      * void get\_nombre\_e\_a\_bis(cJSON\* json, int\* nb\_entites, int\* nb\_associations) : Cette fonction permet de faire le parcours de nœuds de meme niveau pour recuperer le nombre d’entites et d’associations.

2- Module xml

Pour la validation et l’extraction de données, nous avons utilisés la librairie Libxml2.

Validation de données

Nous avons utilisés une seule fonction dans cette partie qui est la suivante:

* + - void xmlIsValid(char \*filename) : La fonction lit un fichier en entrée et vérifie si le fichier est en format xml bien formé en validant le DTD et les caractères xml.

Sa déclaration est dans xmlValideFile.h

Extraction des donnés

Pour cette partie-ci , les fonctions utilisées sont les suivantes :

void extractionData(char \*filename) : La fonction lit un fichier en entrée et initialise un nœud racine et appel la fonction printData pour l’extraction de données

void printData(xmlNode \*a\_node) : C’est au niveau de cette fonction que nous parcourons l’arborescence du fichier xml et que nous extrayons les entites, les attributs,les associations et les relations.

void save\_data(const xmlChar \*data, File \*f) : Cette fonction enregistre la tracce des données extraites dans un fichier nommé saveFile.txt

int nombre\_entite(xmlNode \*root) : La fonction parcours l’arborescence d’un fichier xml et compte le nombre d’entités définit.

void creerEntite(xmlChar \*name) : La fonction dessine dans le fichier svg une entite

Ces fonctions sont déclarées dans extractData.h.