1 Introduction

le manque de données pose un problème crucial au cours de l'implémentation des systèmes d'information. En effet pendant les phases de test les développeurs ont besoin de données afin de pouvoir tester efficacement leurs requêtes SQL.

Notre travail consiste à proposer une solution permettant de générer automatiquement des données pour les tables d'une base de données. Les tables devront être issues d'un schéma relationnel sous format jSON, fourni par l'utilisateur.

Dans un premier temps nous avons proposé le format de description d'un schéma relationnel. Dans un second temps nous avons proposé une implémentation pour la génération automatique de données.

Dans le reste du travail, nous présenterons dans la section 2 le format JSON, dans la section 3 notre solution et dans la section 4 la conclusion de notre travail.

2 Présentation du format json

Définition. JSON signifie JavaScript Object Notation et est un format d'échange de données textuelles.

Il permet de représenter des objets dont les membres peuvent de plusieurs type :

- number ; il permet de représenter des entiers et des nombres à virgule flottante.
- string (chaine de caractères) ; il permet de représenter des chaînes de caractères.
- boolean ; il permet de représenter principalement les valeurs vrai ou faux (true ou false) ;
- object (objet) ;
- array (tableau).

Object Un objet qui est une collection non-ordonnée de paires "nom" : "valeur" entourée par { }.

Array Un tableau est une collection de données de n'importe quel type, séparées par des virgules, le tout entouré par [ ].

La Figure **??** montre un exemple de données présentées sous format JSON. Elle représente un objet JSON contenant les membres `"menu"` et `"commandes"` de type respectifs string et array.

3 Présentation de notre solution

3.1 Format du fichier d'entrée

Nous utilisons un tableau d'objet JSON pour la représentation d'un schéma relationnel, où chaque objet représente le schéma d'une relation.

Chaque

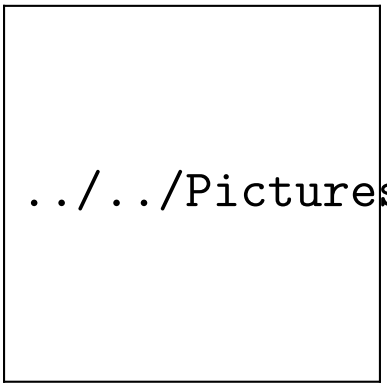

FIGURE 1 – Exemple de fichier au format json

Toutes les relations du Modèle Logique de Données Relationnelles sont représentés dans un tableau([ ]) d'objets ({ }) ; chaque objets représentant ici une relation. Dans chaque objet on a à renseigner donc le nom de la relation ("NomRelation" : "Nom1") et les différents attributs qui sont renseignés ici dans un tableau d'objets ("Attributs" : [{ }, { }, ...]). Chaque attribut sera détaillé dans les objets à travers les clés "name", "type", "options" et/ou "link". Name renseigne juste le nom de l'attribut. Type renseigne le type de l'attribut qui peut être int(entier), string(chaine de caractères), float(les nombres à virgule flottantes) ou boolean (true ou false).

Option qui a comme valeur un objet d'éléments permet d'aller un peu plus en détails à travers les clés "taillemin" (longueur minimum sur lequel cet attribut peut être étendu dans le cas d'un string ou valeur minimum pouvant être prise dans le cas d'un entier), "taillemax" (longueur maximum sur lequel cet attribut peut être étendu dans le cas d'un string ou valeur maximum pouvant être prise dans le cas d'un entier), et/ou "type" (type ici nous spécifie si l'attribut est un nom ou un numéro de téléphone ou mail ou une date dans le cas d'un string), et/ou "serial" (prend la valeur true pour indiquer dans le cas d'un entier que c'est auto incrémenté), et/ou "choix" (cette clé permet à l'utilisateur de spécifier à travers un tableau les valeurs parmi lesquels les occurrences de cet attribut doivent être prises), et/ou "not null" (cette clé prend la valeur true pour spécifier si l'attribut peut prendre ou pas des valeurs nulles), et/ou "unique" (cette clé permet de spécifier si les valeurs générées doivent être uniques ou pas). Link à son tour notifie l'existence d'une dépendance de cet attribut avec un autre d'une autre relation. Sa valeur est un objet ayant comme clés "NomRelation" et "name". NomRelation a comme valeur le nom de la Relation duquel l'attribut dépend. Name renseigne le nom de l'attribut duquel il dépend dans l'autre relation.

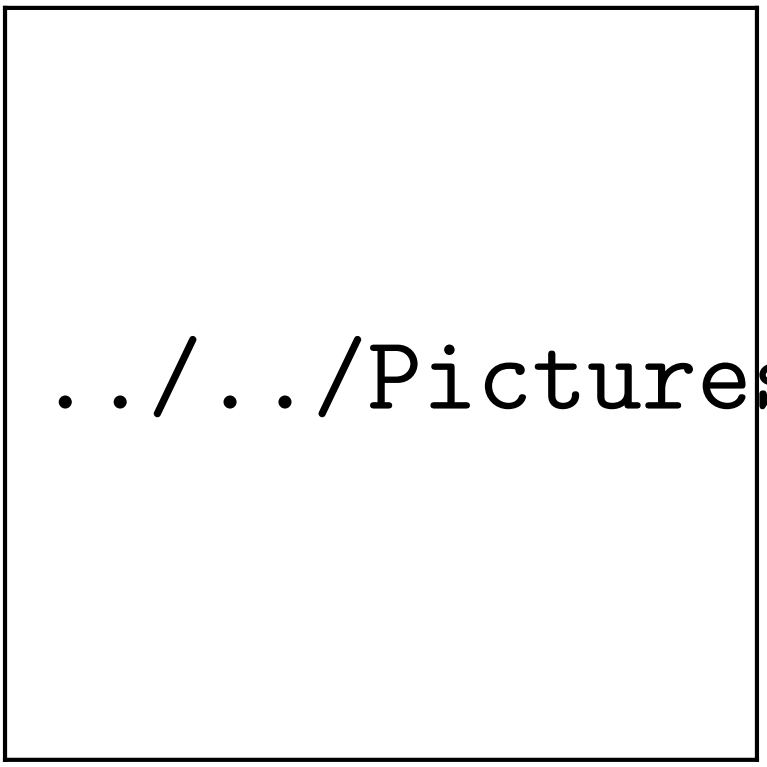

FIGURE 2 – Format du fichier d'entrée

3.2 Implémentation

Nous nous sommes servi de la bibliothèque cJSON pour manipuler le fichier au format JSON. L'implémentation de la solution a conduit à l'élaboration des fonctions qui entre autres permettent de :

- Lire le fichier json envoyé par l'utilisateur, s'assurer qu'il n'y a pas d'erreur syntaxique dans le fichier et le stocker en mémoire ;
- Déterminer l'ordre de génération ;
- Générer des entiers et des floats ;
- Générer des strings en fonction de la nature mails, numéros de téléphone, date, noms ou autres... ;
- Donner la sortie sous le format indiqué par l'utilisateur(txt, json ou sql).

Voici les prototypes de quelques fonctions qui ont été écrites :

- read_file(char *filename) ; //Lis le fichier json.

4 Conclusion