Projet 3° année algorithmique

NoSqlLite

Projet

Le projet consiste à créer un binaire permettant de gérer la persistance de donnée à la manière d'une base de donnée NoSql.

Il n'est pas demandé que le projet soit implémenté sous forme de service via un socket. Il pourra être utilisé uniquement à partir d'une ligne de commande.

Pour simplifier, une base de donnée NoSql est sans schéma, c'est à dire que chaque enregistrement à l'intérieur de la base peut posséder ses propres propriétés. Ainsi peuvent coexister au sein d'une même collection les enregistrements suivants :

```
{
   name : 'Gabriel',
   rate : 12,
   group :'AL2'
}

{
   title : 'SSD 1To CRUCIAL',
   ref : 'SSD-CR-1TO',
   price : '52'
}
```

Par convention, la plupart des bases de donnée NoSql utilisent le formalisme **JSON** pour représenter les modèles et les échanges de de donnée. **Vous devrez suivre cette convention**.

Un enregistrement n'a pas forcement de clé primaire, et une clé primaire n'est pas forcément unique.

Dans le cadre du projet, **vous ne gèrerez que les types suivants** :

- Entier
- Flottants
- Chaine de caractères

Cas d'utilisation à implémenter :

```
$> MySqlLite -collection="student" –insert="{ name : 'Gabriel', rate : 12, group :'AL2'}"
$> MySqlLite -collection="student" –insert="{ name : 'Loic', rate : 18.2, group :'AL2'}"
$> MySqlLite -collection="student" –insert="{ name : 'Benoit', rate : 11, group :'AL1'}"
$> MySqlLite -collection="student" –insert="{ name : 'Jean', rate : 11, group :'AL1'}"

$> MySqlLite -collection="student" –set="{ rate : 14 }" –where="{name :'Benoit'}"

$> MySqlLite -collection="student" –find="{group:'AL1'}" –projection='{name:1}'

Benoit
Jean

$> MySqlLite -collection="student" –find="{}" –projection="{name:1, rate:1}" -sort="{rate:-1}"

Loic          18.2
Benoit        14
Gabriel12
Jean          11

$> Mysql –collection="student" –remove="{ group : 'AL2'}"

$> MySqlLite -collection="student" –find="{}" –projection="{name:1, rate:1}" -sort="{rate:-1}"

Benoit        14
Jean          11
```

($> symbolise l'invite de commande)

Conseils

- Représentez les données en mémoire en utilisant des HashMap.
- Utilisez des algorithmiques de tri avec des comparateurs génériques
- Créez optionnellement des indexes persistants
- Stockez les données dans une structure de fichier simple.
- Implémentez les fonctions suivantes peut vraiment vous aider...
  - t_hashmap* JSON_parse(char* string) ;
    // Convert a JSON String to a HashMap representation.
  - char* JSON_stringify(t_hashmap* map) ;
    // Convert a HashMap to a JSON String representation.
  - void* hashmap_traverse(t_hashmap*map, char* path) ;
    // Traverse a HashMap according to the given path
    // Ex : hashmap_traverse(map, 'student.rate')   14;
  - void hashmap_put(t_hashmap*map, char* path, void* value) ;
    // Traverse a HashMap according to the given path
    // Ex : hashmap_put(map, 'student.rate', 56);
    //   hashmap_traverse(map, 'student.rate')   56

Bon courage.