CS Games 2017

Programmation à relai II

Participants	3
Stations	3
Valeurs	7%
Durée	3 heures

Base de donnée clé-valeur

Nous sommes une start-up de la Silicon Valley et nous voulons implémenter notre propre base de donnée Clé-Valeur pour le "back-end" de notre application, afin de garder en cache les informations les plus souvent accédés. Vous devez implémenter un client ligne de commande pour cette base de donnée, afin de pouvoir tester la possibilité de cette option. Chaque clé est un GUID (Globally Unique Identifier) et chaque valeur est un objet. Les objets peuvent être persisté sous n'importe-quel format. Mais le format d'entrée et de sortie des objets est sous le format JSON.

* La clé peut être autre chose qu'un GUID, mais elle doit être unique dans la base de donnée.

Concept base de donnée clé-valeur

Le premier argument de votre utilitaire de ligne de commande doit être le nom de l'opération exécutée sur la base de données. Les valeurs d'entrées de l'application sont toujours spécifiées en arguments et les retours doivent être imprimés dans le flux standard de sortie. Les opérations suivantes devront être disponible, chaque opération est présentée avec un exemple.

insert : Insérer une nouvelle entrée et retourne la clé nouvellement générée correspondant à cette valeur. Prends en paramètre la valeur à ajouter.
Par exemple, on ajoute cette valeur :

```
{
   "name": "john doe",
   "phoneNumber": "514-555-5555"
}
```

```
$ kvdb insert "{\"name\": \"john doe\", \"phoneNumber\": \"514-555-5555\"}"
```

Et l'utilitaire retourne :

```
{
   "key" : "910bd4c8-34cc-42b4-b434-d4b6ec964f6f"
}
```

insert-with-key : Insérer une nouvelle entrée avec une clé spécifique, si une valeur existe avec cette clé, on écrase l'ancienne valeur.
Par exemple:

```
$ kvdb insert-with-key "{ \"key\": \"910bd4c8-34cc-42b4-b434-d4b6ec964f6f\", \"value\": {
\"name\": \"john doe\", \"phoneNumber\": \"514-555-5555\" } }"
```

L'application ne retourne aucun message après cette opération.
select-by-key : Sélectionner une valeur selon sa clé. On spécifie en argument, le GUID de valeur qu'on veut récupérer. Si elle existe, on retourne la valeur dans le flux. Si aucune entrée n'est trouvé, vous pouvez retourner un objet JSON vide ou rien.

```
$ kvdb select-by-key 910bd4c8-34cc-42b4-b434-d4b6ec964f6f
```

Retourne :

```
{
  "name": "john doe",
  "phoneNumber": "514-555-5555"
}
```

select-by-field : Sélectionner une ou des valeurs avec leur clé, selon la condition passée en argument.

```
$ kvdb select-by-field "name=john doe"
```

Retourne :

```
[{ "key": "910bd4c8-34cc-42b4-b434-d4b6ec964f6f", "value": { "name": "john doe", "phoneNumber": "514-555-5555" } }, ...]
```

delete-by-key : Supprimer une valeur selon sa clé.

```
$ kvdb delete-by-key 910bd4c8-34cc-42b4-b434-d4b6ec964f6f
```

Aucun retour de l'application.

delete-by-field : Supprimer une ou plusieurs valeurs selon un de ces champs. Retourne les clés des valeurs supprimées.

```
$ kvdb delete-by-field "name=john doe"
```

Retourne :

```
["910bd4c8-34cc-42b4-b434-d4b6ec964f6f", ...]
```

update-by-key : Mettre à jour une valeur selon sa clé. Mais n'écrase pas les champs de la valeur s'ils ne sont spécifiés, comparativement à **insert-with-key**.

```
$ kvdb update-by-key "{\"key\": \"910bd4c8-34cc-42b4-b434-d4b6ec964f6f\", \"value\": { \"name\": \"bob\", \"ipAddress\": \"127.0.0.1\" }}"
```

Avec cette opération, nous avons écrasé la valeur du champs "name" qui est maintenant "bob". Par contre, le champs "phoneNumber" existe encore et le champs "ipAddress" est ajouté.
L'application ne retourne rien.
update-by-field : Mettre à jour une valeur ou plusieurs selon un de ces champs. Retourne les clés des valeurs mises-à-jour.

```
$ kvdb update-by-field name=bob "{ \"name\": \"john doe\" }"
```

Retourne :
```
["910bd4c8-34cc-42b4-b434-d4b6ec964f6f", ...]
```

db-info : Retourne les informations sur la base de donnée. Nombre de valeurs dans la base de données et l'espace utilisé en octet.

```
$ kvdb db-info
```

Retourne :
```
{
   "values_count": 1,
   "size": 16
}
```

Fonctionnalité(s)

- ❏ Insertion d'une nouvelle valeur (**insert**) et retour de sa clé, générée par l'application. (2 points)

- ❏ Insertion d'une valeur avec sa clé (**insert-with-key**). Si la clé est déjà utilisé pour une valeur dans la base de donnée, écraser cette valeur par la nouvelle. (3 points)

- ❏ Récupérer une valeur selon sa clé (**select-by-key**). Retourner la valeur dans le flux standard de sortie. (2 points)

- ❏ Retourner toutes les valeurs avec leur clé, qui correspondent à la condition (**select-by-field**). Par exemple: "name=john doe" et retourner la valeur correspondante, non sensible à la casse pour les caractères. Le seul comparateur supporté est l'égalité. Les entrées correspondantes devront être listées avec leur clé et leur valeur. (5 points)

- ❏ Supprimer une valeur la base donnée selon sa clé (**delete-by-key**). (2 points)

- ❏ Supprimer une ou des valeurs selon une condition spécifique (**delete-by-field**). Afficher les clés affectées dans le flux standard de sortie. (3 points)

- ❏ Mettre à jour une valeur selon sa clé (**update-by-key**). (2 points)

- ❏ Mettre à jour une ou des valeurs qui correspondent à la condition (**update-by-field**). L'application doit retourner les clés affectées dans le flux standard de sortie. (5 points)

- ❏ Supporter la concaténation de condition (AND/OR) pour la récupération des valeurs contenus dans la base de données. Par exemple: "name=john doe and phoneNumber=514-555-5555" ou "name=john doe or name=bob". (6 points)

- ❏ Supporter l'opérateur de comparaison x > champs de la valeur, si le champs est un nombre. (3 points)

- ❏ Supporter l'opérateur de comparaison x < champs de la valeur, si le champs est un nombre. (3 points)

- ❏ Supporter l'opérateur x != champs de la valeur, le champs peut-être un nombre ou une chaîne de caractère. (3 points)

- ❏ Implémenter l'opération **db-info** pour retourner les nombres de valeurs entreposées dans la base de données ainsi que la taille en mémoire de la base de données. (2 points)

- ❏ Ajout du support d'un argument d'ordonnancement pour l'opération **select-by-field**. Cette argument permettra d'ordonner la liste selon un champs numérique, en ordre croissant. (5 points)
    - Documenter le fonctionnement de cet argument dans un fichier "lisez-moi".

- ❏ Support des champs imbriqués dans le condition. On peut accéder aux champs imbriqués à l'aide du caractère "." (6 points)

    - Par exemple: On a une valeur :

      ```
      {
         "name": "john doe",
         "phoneNumber": "514-555-5555",
         "address": {
            "civicNumber": 1050,
            "street": "Notre-Dame",
            "city": "Montreal" } ,
      }
      ```

      On peut utiliser la condition "address.street=Notre-Dame" pour récupérer cette valeur.

- ❏ Supporter la fonction “startswith” pour les conditions de sélection. (3 points)
    - Par exemple:

    $ kvdb select-by-field startswith(name, ‘john’)

    Retourne:
    [{ "key": "910bd4c8-34cc-42b4-b434-d4b6ec964f6f", "value": { "name": "john doe", "phoneNumber": "514-555-5555" } }, ...]

- ❏ Chiffrer le fichier qui contient le contenu de la base de donnée. (2 points)

- ❏ Inclure un fichier “lisez-moi” dans votre solution, documentant les fonctionnalités complétées ainsi que le lancement de votre application. (1 point)

- ❏ Inclure un ficher “run.sh” pour démarrer votre solution. (1 point)