Lab 2 Bases de Dados de Documentos

Objetivos

Os objetivos deste trabalho são:

- Compreender os fundamentos das bases de dados baseadas em documentos.
- Instalar e utilizar uma solução de utilização livre - mongodb.
- Desenvolver soluções para diversos casos de uso

Nota prévia

Este módulo deverá ser preferencialmente desenvolvido em Linux. Caso pretenda usar Windows verifique as notas sobre compatibilidade do software que irá usar.
Submeta o código/resultados/relatórios no elearning. Utilize uma pasta (1, 2, ..) para cada exercício, compactadas num único ficheiro.
Bom trabalho!

2.1 MongoDB – Instalação e exploração por linha de comandos

MongoDB é uma base de dados orientada a documentos representados numa estrutura JSON (internamente usa BSON, uma versão binária de JSON). É um projeto de código aberto, com licença GNU AGPL (Affero General Public License).

a) Instale o MongoDB no seu computador pessoal (https://www.mongodb.com/) e execute o servidor (`mongod`). Por exemplo (pode variar consoante o SO):

```
$ mongod --dbpath <path to data directory>
```

b) Estude o funcionamento do sistema testando os comandos mais usados, através de linha de comandos (programa cliente `mongo`):

```
$ mongo [--username <user> --password <pass> --host <host> --port <portN>]
```

Também pode usar uma aplicação de gestão interativa como, por exemplo, o Studio 3T ou o Mongo Compass.

Consulte os slides disponibilizados para a disciplina e sítios web com documentação sobre o MongoDB. Alguns exemplos:

- *MongoDB Docs, https://docs.mongodb.com*
- *https://www.tutorialspoint.com/mongodb/*

Deve estudar conceitos e funcionalidades tais como:

- Estrutura de armazenamento (DB, Collections, Documents)
- JSON e Javascript
- Escrita, Leitura, Edição, Remoção (CRUD)
- Tipos e arrays

- Índices
- Agregações e mapreduce

c) Produza um bloco de notas, `CBD_L201_<NMEC>.txt`, com os seus apontamentos pessoais sobre esta fase, incluindo todas as iterações com o mongo, comandos e algumas linhas de resultados. Exemplo:

```
# CBD – Lab201 – <Nome>

> show dbs
admin   0.000GB
cbd     0.001GB
config  0.000GB
local   0.000GB
sales   0.000GB

> ...
```

2.2 MongoDB – Construção de queries

Para este exercício deverá utilizar o ficheiro "restaurantes.json". Este, contém informação sobre 3772 restaurantes, de acordo com a seguinte estrutura JSON:

```
{
    "address": {
       "building":"1007",
       "coord":[
           -73.856077,        // assuma como latitude
           40.848447          // assuma como longitude
       ],
       "rua":"Morris Park Ave",
       "zipcode":"10462"
   },
   "localidade":"Bronx",
   "gastronomia":"Bakery",
   "grades":[
         { "date": { "$date": 1393804800000 }, "grade": "A", "score": 2 },
         { "date": { "$date": 1378857600000 }, "grade": "A", "score": 6 },
         { "date": { "$date": 1358985600000 }, "grade": "A", "score": 10 },
         { "date": { "$date": 1322006400000 }, "grade": "A", "score": 9 },
         { "date": { "$date": 1299715200000 }, "grade": "B", "score": 14 }
    ],
    "nome":"Morris Park Bake Shop",
   "restaurant_id":"30075445"
}
```

Poderá colocar estes dados na sua instalação local do mongo, usando o comando seguinte:

```
$ mongoimport --db cbd --collection restaurants --drop --file
<path/>restaurants.json
```

Execute o cliente mongo (command line) e verifique se os dados foram carregados no servidor.

```
$ mongo

> use cbd
> show collections
restaurants
> db.restaurants.count()
3772
```

Usando o mongo em modo de linha de comandos, ou um programa cliente, escreva expressões/queries para obter os resultados esperados para as perguntas seguintes. Escreva todas as respostas no ficheiro `CBD_L202_<NMEC>.txt`, onde `<NMEC>` deve ser substituído pelo seu nº mecanográfico. Para cada pergunta, escreva a pergunta, o comando utilizado e o resultado. Se pretender incluir comentários use "`//`".

Exemplo:

```
// NMEC: 12345

// 1. Liste todos os documentos da coleção.
db.restaurants.find()
// 3772

// 2. Apresente os campos restaurant_id, nome, localidade e gastronomia para
todos os documentos da coleção
...
```

Queries:

1. Liste todos os documentos da coleção.
2. Apresente os campos *restaurant_id*, *nome*, *localidade* e *gastronomia* para todos os documentos da coleção.
3. Apresente os campos *restaurant_id*, *nome*, *localidade* e código postal (*zipcode*), mas exclua o campo *_id* de todos os documentos da coleção.
4. Indique o total de restaurantes localizados no Bronx.
5. Apresente os primeiros 15 restaurantes localizados no Bronx, ordenados por ordem crescente de nome.
6. Liste todos os restaurantes que tenham pelo menos um *score* superior a 85.
7. Encontre os restaurantes que obtiveram uma ou mais pontuações (*score*) entre [80 e 100].
8. Indique os restaurantes com latitude inferior a -95,7.
9. Indique os restaurantes que não têm *gastronomia* "American", tiveram uma (ou mais) pontuação superior a 70 e estão numa latitude inferior a -65.
10. Liste o *restaurant_id*, o *nome*, a *localidade* e *gastronomia* dos restaurantes cujo nome começam por "Wil".
11. Liste o *nome*, a *localidade* e a *gastronomia* dos restaurantes que pertencem ao Bronx e cuja *gastronomia* é do tipo "American" ou "Chinese".
12. Liste o *restaurant_id*, o *nome*, a *localidade* e a *gastronomia* dos restaurantes localizados em "Staten Island", "Queens", ou "Brooklyn".

13. Liste o *nome*, a *localidade*, o *score* e *gastronomia* dos restaurantes que alcançaram sempre pontuações inferiores ou igual a 3.
14. Liste o *nome* e as avaliações dos restaurantes que obtiveram uma avaliação com um *grade* "A", um *score* 10 na data "2014-08-11T00: 00: 00Z" (ISODATE).
15. Liste o *restaurant_id*, o *nome* e os *score* dos restaurantes nos quais a segunda avaliação foi *grade* "A" e ocorreu em ISODATE "2014-08-11T00: 00: 00Z".
16. Liste o *restaurant_id*, o *nome*, o *endereço* (*address*) e as coordenadas geográficas (*coord*) dos restaurantes onde o 2º elemento da matriz de coordenadas tem um valor superior a 42 e inferior ou igual a 52.
17. Liste *nome, gastronomia e localidade* de todos os restaurantes ordenando por ordem crescente da *gastronomia* e, em segundo, por ordem decrescente de *localidade*.
18. Liste *nome*, *localidade, grade* e *gastronomia* de todos os restaurantes localizados em Brooklyn que não incluem *gastronomia* "American" e obtiveram uma classificação (*grade*) "A". Deve apresentá-los por ordem decrescente de *gastronomia*.
19. Indique o número total de avaliações (*numGrades*) na coleção.
20. Apresente o *nome* e número de avaliações (*numGrades*) dos 3 restaurante com mais avaliações.
21. Apresente o número total de avaliações (numGrades) em cada dia da semana.
22. Conte o total de restaurante existentes em cada localidade.
23. Indique os restaurantes que têm *gastronomia* "Portuguese", o somatório de *score* é superior a 50 e estão numa latitude inferior a -60.
24. Apresente o número de gastronomias diferentes na rua "Fifth Avenue"
25. Apresente o *nome* e o score médio (*avgScore*) e número de avaliações (*numGrades*) dos restaurantes com score médio superior a 30 desde 1-Jan-2014.
26. .. 30. Descreva 5 perguntas adicionais à base dados (alíneas 26 a 30), significativamente distintas das anteriores, e apresente igualmente a solução de pesquisa para cada questão.

2.3 MongoDB – Driver (Java)

Para este exercício deverá utilizar a coleção *restaurantes*, mas agora acedendo programaticamente através de um dos drivers disponibilizados no sítio do mongo (https://docs.mongodb.com/ecosystem/drivers/).
Deverá usar o driver de Java criando um projeto *maven*, ou utilizando os seguintes ficheiros jar: monogodb driver core, mongodb driver, bson.

a) Desenvolva um programa simples que permita testar inserção, edição e pesquisa de registos sobre a coleção.

b) Crie índices: um para localidade; outro para gastronomia; e um de texto para o nome. Use pesquisas para testar o funcionamento e o verifique o desempenho (*como são poucos documento os resultados poderão não melhorar*).

c) Selecione 5 perguntas/comandos do exercício 2.2 e reimplemente-os em Java.

d) Construa e teste os seguintes métodos, apresentando o resultado da sua execução

no ficheiro `CBD_L203_<NMEC>.txt`:

- `public int countLocalidades()`

```
Numero de localidades distintas: 6
```

- `Map<String, Integer> countRestByLocalidade()`

```
Numero de restaurantes por localidade:
 -> Queens - 738
 -> Staten Island - 158
 -> Manhattan - 1883
 -> Brooklyn – 684
...
```

- `List<String> getRestWithNameCloserTo(String name)`

```
Nome de restaurantes contendo 'Park' no nome:
 -> Morris Park Bake Shop
 -> New Park Pizzeria & Restaurant
 -> Parkside Restaurant
 -> New Parkway Restaurant
...
```

2.4 Sistema de Atendimento

Este exercido é semelhante ao do laboratório passado. Tenha em atenção o desempenho da solução proposta.

a) Construa um sistema de atendimento em Java usando Mongo e os respetivos drivers. Este deverá permitir que determinado utilizador (*username*) peça um número máximo (*limit*) de produtos (*product*) por unidade de tempo (*timeslot*). O sistema deve ir registando os pedidos (username, product), não atendendo pedidos de novos produtos se já tiver excedido o limite máximo definido para a janela temporal. Caso exceda o limite, o sistema deve produzir uma mensagem de erro. O *limit* e *timeslot* são valores definidos para todo o sistema.

   Exemplo: O sistema atende, para cada utilizador, um máximo de 30 produtos diferentes a cada 60 minutos.

b) Altere o programa para que cada pedido possa ter uma quantidade (quantity) associada, i.e. as unidades de produto solicitados no pedido. Nesta versão, o limite passa a ser o número total de unidades de produto requisitadas, por janela temporal.

   Exemplo: O sistema atende, para cada utilizador, um máximo de 30 unidades de produtos a cada 60 minutos.

c) Comente, sucintamente, as diferenças conceptuais entre as implementações baseadas em Redis e Mongo. Evidencie quais os prós e contras das duas soluções e qual se adequa melhor à solução deste problema. Execute operações (read/write) similares nas duas soluções (Redis e MongoDB), extraindo métricas de desempenho, fazendo uma análise crítica dos resultados.

Deverá entregar os ficheiros:

`*.java` – dos dois exercícios

`CBD_L204a-out_<NMEC>.txt` - exemplos de interação e de output do programa a)

`CBD_L204b-out_<NMEC>.txt` – exemplos de interação e de output do programa b)

`CBD_L204c_<NMEC>.txt` – Resposta à pergunta c)

2.5 MongoDB – Funções do lado do servidor

Neste exercício pretende-se utilizar e desenvolver funções javascript para executar no servidor do MongoDB.

a) Copie o ficheiro *populatePhones.js* para uma pasta da sua área de trabalho. Arranque o cliente mongo e descarregue no servidor a função "`populatePhones`". Analise e teste o funcionamento desta função.

```
> load("<pasta da sua área de trabalho>/populatePhones.js")
true

> populatePhones
function (country, start, stop) {

  var prefixes = [21, 22, 231, 232, 233, 234 ];
  for (var i = start; i <= stop; i++) {

    var prefix = prefixes[(Math.random() * 6) << 0]
    var countryNumber =
      (prefix * Math.pow(10, 9 - prefix.toString().length)) + i;
    var num = (country * 1e9) + countryNumber;
    var fullNumber = "+" + country + "-" + countryNumber;

    db.phones.insert({
      _id: num,
      components: {
        country: country,
        prefix: prefix,
        number: i
      },
      display: fullNumber
    });
    print("Inserted number " + fullNumber);
  }
  print("Done!");
}

> populatePhones(351, 1, 5)
Inserted number +351-233000001
Inserted number +351-231000002
Inserted number +351-234000003
Inserted number +351-234000004
Inserted number +351-220000005
Done!
```

(Nota: carrega na "current shell". Não armazena no servidor)

b) Depois dos testes da alínea anterior, limpe a coleção (`db.phones.drop()`) e, usando esta função, crie 200,000 números, por exemplo:

```
> populatePhones(351, 1, 200000)
```

(Nota: esta operação não é imediata).

No final, verifique o conteúdo da coleção (usando as funções `find, count, ...`).

c) Construa uma função/expressão que conte o número de telefones existentes em cada um dos indicativos nacionais (*prefix*).

d) Construa, e teste no servidor, uma função em JavaScript que encontre um tipo de padrão na lista (e.g., capicuas, sequências, dígitos não repetidos, etc.).

2.6 Base de Dados com Temática Livre (Opcional)

Este exercício tem como objetivo a criação de uma coleção que tire partido do modelo de dados de MongoDB. Tenha em atenção as seguintes recomendações e requisitos:

(a) A base de dados pode ser criada por adaptação ou importação de um dataset público.

(b) O número de documentos da base de dados e a complexidade da estrutura de dados de cada documento é um fator a ter em conta, para poder criar queries com alguma complexidade e diversidade de operações lógicas. Por exemplo, criar uma base de dados com centenas de documentos e utilizar "array of embedded documents".

(c) Crie 6 queries expressivas do seu domínio de conhecimento utilizando o operador *find({...}, {...})*.

(d) Crie 6 queries expressivas do seu domínio de conhecimento utilizando o operador *aggregate ($group, $project, $unwind, $match, etc)*.

Nota: nas alíneas c) e d) podem ser feitas em scripts mongo ou usando a API java.
Não deve replicar queries disponíveis em fóruns públicos.
Descreva a estrutura do dataset e todas as perguntas-respostas no ficheiro `CBD_L205_<NMEC>.TXT`.