Ingeniería en Sistemas Computacionales

Bases de Datos No Relacionales

MongoDB Lab 3 - Índices y parametrización de queries

Marco Ricardo Cordero Hernández

Tlaquepaque, Jal., 26 de marzo de 2023

Después de haber modificado los archivos correspondientes de forma que aceptaran las acciones de modificación y eliminación de documentos, se ha solicitado la expansión de la funcionalidad de búsqueda de los mismo, permitiendo ahora parámetros como búsqueda por puntuación mínima y demás parámetros numéricos y textuales.

Con los módulos previamente integrados, el texto auxiliar que ofrece el cliente quedaría como el siguiente.

```
usage: client.py [-h] [-i ID] [-r RATING] [-p PAGES] [-rc RATINGS_COUNT] [-t TITLE] [-l LIMIT] [-s SKIP] [-j JSON] {search,get,update,delete}

positional arguments:
  {search,get,update,delete}
                        Action to be used for the books library

options:
  -h, --help            show this help message and exit
  -i ID, --id ID        Provide a book ID which related to the book action
  -r RATING, --rating RATING
                        Search parameter to look for books with average rating equal or above the param (0 to 5)
  -p PAGES, --pages PAGES
                        Provide a number of page search parameter (comparison operators supported)
  -rc RATINGS_COUNT, --ratings-count RATINGS_COUNT
                        Provide a ratings count search parameter (comparison operators supported)
  -t TITLE, --title TITLE
                        Provide a title search parameter (behaves as LIKE operator); Wrap search term between quotes for multiple words
  -l LIMIT, --limit LIMIT
                        Limit results length with given boundary
  -s SKIP, --skip SKIP  Skip given number of results
  -j JSON, --json JSON  JSON file parameter for book updating
```

Como se puede apreciar, no solo se han implementado las funciones de límite y omisión de documentos, sino que también se ha añadido la función extendida de operadores de comparación, tales como >, >=, <, <=, != e ==. La manera en la que esto funciona es mediante la declaración de un valor numérico seguido del identificador del operador, por ejemplo, “300gte”, lo cual indica que se buscarán documentos al cual le corresponde un campo con valores mayores o iguales a 300, independientemente de lo que representen. Posteriormente, esta entrada es procesada e interpretada en el archivo en donde se definen las rutas de la aplicación.

El siguiente código muestra el comportamiento del endpoint principal de este laboratorio:

```python
@router.get("/", response_description="Get all books", response_model=List[Book])
def list_books(request: Request, rating: float = 0, num_pages: str = '', ratings_count: str = '', title: str = '',
limit: int = 10, skip: int = 0):
    search_params = {}

    allowed_operators = ('eq', 'gt', 'gte', 'lt', 'lte', 'ne')
    locals_vals = zip(list(locals().keys())[2:-4], list(locals().values())[2:-4])

    for k, v in locals_vals:
        param_re = re.search(r'(\d+)(.*)', v)

        if (param_re):
            if (not param_re.group(2)):
                search_params[k] = float(param_re.group(1))
            else:
                if (param_re.group(2) not in allowed_operators):
                    print(f'**** Operator "{param_re.group(2)}" not allowed')
                else:
                    search_params[k] = {f'${param_re.group(2)}': float(param_re.group(1))}
        elif (v):
            search_params[k] = {'$regex': v}

    books = list(request.app.database["books"]
                 .find(search_params)
                 .skip(skip)
                 .limit(limit))
    return books
```

El código adicional se encuentra dentro del mismo contenedor en el que se entrega este laboratorio.

Finalmente, a manera de refuerzo de conocimientos, se han creado los siguientes índices dentro de mongosh:

```
iteso> db.books.createIndex({"num_pages": 1})
num_pages_1
iteso> db.books.createIndex({"ratings_count": 1})
ratings_count_1
iteso> db.books.createIndex({"title": "text"})
title_text
```

Estos índices han sido generados con el fin de darle una velocidad de consulta acelerada cuando se incluyan estos parámetros en las mismas. También, al contar con un índice de texto, se pueden extraer documentos con subcadenas contenidas dentro de los atributos de forma más eficiente,