

Instituto Tecnológico de Buenos Aires

TRABAJO PRÁCTICO

ENTREGA TPO BASE DE DATOS II (72.41)

Grupo 2

Integrantes:

Santiago Tomás Medin - Legajo N° 62076 (smedin@itba.edu.ar)
Bruno Enzo Baumgart - Legajo N° 62161 (bbaumgart@itba.edu.ar)
Santiago José Hirsch - Legajo N° 62169 (shirsch@itba.edu.ar)

Docentes:

Guillermo Rodríguez Cecilia Rodriguez Babino

Tabla de contenidos

| Tabla de contenidos | 2 |
|-----------------------------------|----|
| Ejercicio 1 - MongoDB | 3 |
| Importación de datasets y scripts | 3 |
| Item a | 3 |
| Item b | 3 |
| Item c | 4 |
| Item d | 5 |
| Ejercicio 2 - Neo4j | 6 |
| Configuración BD | 6 |
| Generación BD | 6 |
| Carga BD | 6 |
| Creación de relaciones | 7 |
| Item a | 7 |
| Item b | 8 |
| Item c | 8 |
| Item d | 9 |
| Ejercicio 3 - Redis | 10 |
| Importación de datasets y scripts | 10 |
| Item a | 10 |
| Item b | 11 |
| Item c | 11 |
| Item d | 12 |
| Item e | 12 |

Ejercicio 1 - MongoDB

Importación de datasets y scripts

Para importar el csv albumlist.csv y los scripts necesarios para la resolución del ejercicio se ejecuta el script exercise1_setup.sh que se encuentra dentro del directorio scripts/exercise1 (una vez ya creada la conexión al contenedor y desde una terminal local):

1. Le otorgamos permisos de ejecución al script:

chmod u+x ./scripts/exercise1/exercise1_setup.sh

2. Ejecutamos el script

./scripts/exercise1/exercise1_setup.sh

3. Le otorgamos permisos de ejecución a los scripts importados (desde el Shell bash del contenedor):

chmod u+x exercise1_*.sh

Item a

Importe el archivo albumlist.csv (o su versión RAW) a una colección. Este archivo cuenta con el top 500 de álbumes musicales de todos los tiempos según la revista Rolling Stones.

Para resolver esto se ejecutó el siguiente comando:

```
mongoimport --db musicDB --collection albums --type csv --headerline --file albumlist.csv
```

Item b

Cuente la cantidad de álbumes por año y ordénelos de manera descendente (mostrando los años con mayor cantidad de álbumes al principio).

Para resolver esto se ejecutó la siguiente consulta:

```
}
},
{
    $sort: { count: -1 }
}
]).forEach(printjson);
```

La misma produce la siguiente salida (se muestran los primeros 5 pero en la salida real se muestran todos los años con sus respectivas cantidades):

```
{
    _id: 1970,
    count: 78
}
{
    _id: 1972,
    count: 72
}
{
    _id: 1973,
    count: 69
}
{
    _id: 1969,
    count: 66
}
{
    _id: 1968,
    count: 63
}
```

Item c

A cada documento, agregarle un nuevo atributo llamado 'score' que sea 501-Number.

Para resolver esto se ejecutó la siguiente sentencia:

```
db.albums.find().forEach(function(album) {
   var rank = album.Number;
   var score = 501 - rank;
   db.albums.updateOne(
        { _id: album._id },
        { $set: { score: score } }
   );
});
```

Item d

Realice una consulta que muestre el 'score' de cada artista.

Para resolver esto se ejecutó la siguiente consulta:

La misma produce la siguiente salida (se muestran los primeros 5 pero en la salida real se muestran todos los artistas con sus respectivos puntajes):

```
{
    _id: 'The Beatles',
    Score: 11565
}
{
    _id: 'The Rolling Stones',
    Score: 10812
}
{
    _id: 'Bob Dylan',
    Score: 10131
}
{
    _id: 'Bruce Springsteen',
    Score: 6753
}
{
    _id: 'The Who',
    Score: 6630
}
```

Ejercicio 2 - Neo4j

Configuración BD

Generación BD

Para generar la base de grafos en cuestión, ejecutar el comando desde la interfaz web de Neo4J:

```
:play northwind-graph
```

Carga BD

Para cargar la base de datos generada ejecutar los siguientes comandos:

1. Carga de los productos

```
LOAD CSV WITH HEADERS FROM "https://data.neo4j.com/northwind/products.csv"

AS row

CREATE (n:Product)

SET n = row,

n.unitPrice = toFloat(row.unitPrice),

n.unitsInStock = toInteger(row.unitsInStock), n.unitsOnOrder = toInteger(row.unitsOnOrder),

n.reorderLevel = toInteger(row.reorderLevel), n.discontinued = (row.discontinued <> "0")
```

2. Carga de las categorías

```
LOAD CSV WITH HEADERS FROM "https://data.neo4j.com/northwind/categories.csv"

AS row

CREATE (n:Category)

SET n = row
```

3. Carga de los proveedores

```
LOAD CSV WITH HEADERS FROM "https://data.neo4j.com/northwind/suppliers.csv"

AS row

CREATE (n:Supplier)

SET n = row
```

Creación de relaciones

Para crear las relaciones entre los nodos de la base de datos ejecutar los siguientes comandos:

1. Relación PART_OF

Con el siguiente comando creamos la relación *PART_OF* entre los **productos** y las **categorías**. En donde un **producto** es *PART_OF* de una **categoría**.

```
MATCH (p:Product),(c:Category)
WHERE p.categoryID = c.categoryID
CREATE (p)-[:PART_OF]->(c)
```

2. Relación SUPPLIES

Con el siguiente comando creamos la relación *SUPPLIES* entre los **productos** y los **proveedores**. En donde un **proveedores** *SUPPLIES* uno o más **productos**.

```
MATCH (p:Product),(s:Supplier)
WHERE p.supplierID = s.supplierID
CREATE (s)-[:SUPPLIES]->(p)
```

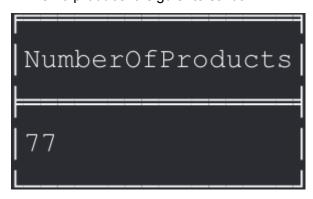
Item a

¿Cuántos productos hay en la base?

Para responder esta consulta se ejecuta el siguiente comando en la interfaz de Neo4j:

```
match (p:Product)
return count(p) as NumberOfProducts
```

El mismo produce la siguiente salida:



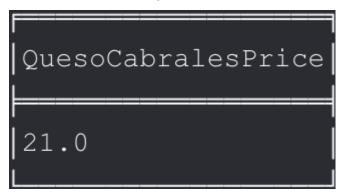
Item b

¿Cuánto cuesta el "Queso Cabrales"?

Para responder esta consulta se ejecuta el siguiente comando en la interfaz de Neo4j:

```
match (p:Product {productName: "Queso Cabrales"})
return p.unitPrice as QuesoCabralesPrice
```

El mismo produce la siguiente salida:



Item c

¿Cuántos productos pertenecen a la categoría "Condiments"?

Para responder esta consulta se ejecuta el siguiente comando en la interfaz de Neo4j:

```
match (p:Product)-[:PART_OF]->(c:Category {categoryName:"Condiments"})
return count(p) as CondimentsProductsCount
```

El mismo produce la siguiente salida:



Item d

Del conjunto de productos que ofrecen los proveedores de "UK", ¿Cuál es el nombre y el precio unitario de los tres productos más caros?

Para responder esta consulta se ejecuta el siguiente comando en la interfaz de Neo4j:

```
match (s:Supplier {country:"UK"})-[:SUPPLIES]->(p:Product)
return p.productName as ProductName, p.unitPrice as UnitPrice
order by p.unitPrice
desc limit 3
```

El mismo produce la siguiente salida:

| ProductName | UnitPrice |
|---------------------|--------------------|
| "Chang" | 19.0 |
| "Chai" | 18.0 |
| "Aniseed Syrup" | 10.0 |

Ejercicio 3 - Redis

Importación de datasets y scripts

Para importar el csv bataxi.csv y el script exercise3.py al contenedor se ejecuta el script exercise3_setup.sh que se encuentra dentro del directorio scripts/exercise3 (una vez ya creada la conexión al contenedor y desde una terminal local):

1. Le otorgamos permisos de ejecución al script:

```
chmod u+x ./scripts/exercise3/exercise3_setup.sh
```

2. Ejecutamos el script

```
./scripts/exercise3/exercise3_setup.sh
```

3. Nos dirigimos al directorio en donde se encuentran los datasets y scripts (desde el Shell bash del contenedor):

```
cd ..
```

4. Le otorgamos permisos de ejecución al script:

```
chmod u+x ./exercise3.py
```

Dentro del script *exercise3.py* se utiliza el paquete de Python *Redis* para conseguir la conexión a la BD y para hacer las consultas. La siguiente es la definición de la bd dentro del script:

```
redis_db = redis.StrictRedis(host='localhost', port=6379, db=0)
```

Item a

Importar los datos del archivo a Redis

Para lograr esto se utilizó el siguiente método:

```
def item_a(csv_file):
    print("a. Importar los datos del archivo a Redis")
    with open(csv_file, 'r', encoding='utf-8-sig') as file:
        reader = csv.DictReader(file)
```

```
for row in reader:
    id_viaje = row['id_viaje_r']
    origen_latitud = float(row['origen_viaje_x'])
    origen_longitud = float(row['origen_viaje_y'])
    redis_db.geoadd("bataxi", (origen_latitud, origen_longitud, id_viaje))
print("Datos importados correctamente\n\n")
```

Item b

¿Cuántos viajes se generaron a 1 km de distancia de estos 3 lugares?

Para lograr esto se utilizó el siguiente método (el array *places* fue provisto por la cátedra en la consigna):

El mismo produce la siguiente salida:

¿Cuántos viajes se generaron a 1 km de distancia de estos 3 lugares? Parque Chas, UTN y ITBA Madero Viajes a 1 km de Parque Chas: 339 Viajes a 1 km de UTN: 9 Viajes a 1 km de ITBA Madero: 242

Item c

¿Cuántas KEYS hay en la base de datos Redis?

Para lograr esto se utilizó el siguiente método:

```
def item_c():
    print("c. ¿Cuántas KEYS hay en la base de datos Redis?")
    keys_count = len(redis_db.keys('*'))
    print(f"Número de keys en Redis: {keys_count}\n\n")
```

El mismo produce la siguiente salida:

¿Cuántas KEYS hay en la base de datos Redis? Número de keys en Redis: 1

Item d

¿Cuántos miembros tiene la key 'bataxi'?

Para lograr esto se utilizó el siguiente método:

```
def item_d():
    print("d. ¿Cuántos miembros tiene la key 'bataxi'?")
    members_count = redis_db.zcard('bataxi')
    print(f"Número de miembros en 'bataxi': {members_count}")
```

El mismo produce la siguiente salida:

¿Cuántos miembros tiene la key 'bataxi'? Número de miembros en 'bataxi': 19148

Item e

GEOADD utiliza la estructura de datos de Redis llamada 'sorted set'.