25-12-2024

Bases de datos no relacionales

PRA1

Juan Luis Acebal Rico
GRADO DE CIENCIA DE DATOS APLICADA

Indice

Ejercicio 1. Cassandra

1.

```
--Lo primero, desde la terminal entro usando cqlsh

--Creo el keyspace, selecciono usar CRIMES

CREATE KEYSPACE CRIMES WITH REPLICATION = { 'class': 'SimpleStrategy', 'replication_factor': 1 };

USE CRIMES

--Elimino tabla si existe

DROP TABLE IF EXISTS TESTING_AGGREGATE;

--Creo la tabla

CREATE TABLE TESTING_AGGREGATE (

   COLUMNA1 TEXT,

   COLUMNA2 TEXT,

   COLUMNA3 TEXT,

   COLUMNA4 TEXT,

   PRIMARY KEY (COLUMNA1, COLUMNA2, COLUMNA3)

);

DESCRIBE TABLES;
```

testing_aggregate

```
-- Cargo los datos desde el archivo TESTING_AGGREGATE.csv a la tabla TESTING_AGGREGATE

COPY TESTING_AGGREGATE (COLUMNA1, COLUMNA2, COLUMNA3, COLUMNA4)

FROM '/home/uoclabs/data/TESTING_AGGREGATE.csv'

WITH DELIMITER=',' AND HEADER=TRUE;
```

Using 1 child processes

Starting copy of crimes.testing_aggregate with columns [columna1, columna2, columna3, columna4].

Processed: 2 rows; Rate: 3 rows/s; Avg. rate: 5 rows/s

2 rows imported from 1 files in 0.410 seconds (0 skipped).

SELECT columna1, columna2,columna3, columna4 FROM TESTING_AGGREGATE;

```
 columna1 | columna2 | columna3 | columna4
----------+----------+----------+----------
 clave1_1 | clave2_1 | clave3_1 |   valor1
 clave1_2 | clave2_2 | clave3_2 |   valor2

(2 rows)
cqlsh:crimes>
```

Consulta 1.1

Inserto el nuevo registro con insert y a continuacion hago un select

```
INSERT INTO TESTING_AGGREGATE (COLUMNA1, COLUMNA2, COLUMNA3,
COLUMNA4)
VALUES ('clave1_1', 'clave2_1', 'clave3_1', 'valor3');
```

```
cqlsh:crimes> SELECT * FROM TESTING_AGGREGATE;

 columna1 | columna2 | columna3 | columna4
----------+----------+----------+----------
 clave1_1 | clave2_1 | clave3_1 |   valor3
 clave1_2 | clave2_2 | clave3_2 |   valor2

(2 rows)
cqlsh:crimes>
```

Lo que vemos aqui es que ha insertado “un nuevo registro”, sobreescribiendo el existente, ya que la clave primaria que es las 3 primeras columnas, ya existia. Una base de datos relacional depende, en general no tiene que haber problemas de consistencia de datos si el modelado se ha hecho bien, si esta definida como clave primaria y unica, da error, si no es unica, dependiendo de la base de datos puede o no hacer nada, eliminarlo, o sobreescribirlo, aunque en general, esta mejor hecha la parte de inconsistencia en SQL que en no sql, por tanto en no sql tenemos que hacer mas atencion, para evitar inconsistencias.

Proceso de creacion de tablas y copia de datos

Ahora voy a crear las distintas tablas que voy a utilizar, sabiendo las limitaciones de cassandra respecto a la :

--AREA_DESC_AGE_SEX (NO LA USE FINALMENTE)

```
DROP TABLE IF EXISTS AREA_SUB_AREA_DESC_AGE_SEX;
CREATE TABLE AREA_SUB_AREA_DESC_AGE_SEX (

   area_name TEXT,

   sub_area TEXT,

   crime_description TEXT,

   vict_age INT,

   vict_sex TEXT,

   total_count INT,

   PRIMARY KEY ((area_name), sub_area, crime_description ,total_count)

);
```

--Me parece coherente que la primary key sea la suma de esas columnas ya que un crimen tiene como minimo esas combinaciones, y puesto que no tenemos la fecha, hora y lugar, puede existir la situacion que, como hemos visto en el apartado anterior, sobreescriba otros registros.

--Copio el csv

```
COPY AREA_SUB_AREA_DESC_AGE_SEX (area_name, sub_area, crime_description, vict_age,
vict_sex, total_count)

FROM '/home/uoclabs/data/AREA_SUB_AREA_DESC_AGE_SEX.csv'

WITH DELIMITER=',' AND HEADER=TRUE;
```

Using 1 child processes

Starting copy of crimes.area_sub_area_desc_age_sex with columns [area_name, sub_area, crime_description, vict_age, vict_sex, total_count].
Processed: 529552 rows; Rate: 5225 rows/s; Avg. rate: 6971 rows/s
529552 rows imported from 1 files in 0 day, 0 hour, 1 minutes, and 15.964 seconds (0 skipped).
cqlsh:crimes>

--el csv tiene con la cabecera 529553 lineas, parece correcto

--WEAPON (CONSULTA 2 Y 4)

```
--Segunda tabla, hago igual que la primera, hago una clave primaria combinada

DROP TABLE IF EXISTS YEAR_MONTH_DAY_TIME_AREA_SUB_AREA_WEAPON;

CREATE TABLE YEAR_MONTH_DAY_TIME_AREA_SUB_AREA_WEAPON (

   year_occ INT,

   month_occ INT,

   day_occ INT,

   time_occ TEXT,

   sub_area TEXT,

   weapon_desc TEXT,

   total_count INT,

      PRIMARY KEY (weapon_desc, year_occ, month_occ, day_occ, time_occ, sub_area));

--copio

COPY YEAR_MONTH_DAY_TIME_AREA_SUB_AREA_WEAPON (year_occ, month_occ, day_occ,
time_occ, sub_area, weapon_desc, total_count)
```

```
FROM '/home/uoclabs/data/YEAR_MONTH_DAY_TIME_AREA_SUB_AREA_WEAPON.csv'
WITH DELIMITER=',' AND HEADER=TRUE;
```

Using 1 child processes

Starting copy of crimes.year_month_day_time_area_sub_area_weapon with columns [year_occ, month_occ, day_occ, time_occ, sub_area, weapon_desc, total_count].
Processed: 869903 rows; Rate: 4664 rows/s; Avg. rate: 6215 rows/s
869903 rows imported from 1 files in 0 day, 0 hour, 2 minutes, and 19.980 seconds (0 skipped).
cqlsh:crimes>

--el csv tiene con la cabecera 869904 lineas, parece correcto

--DATE_TIME_AREA_AGE_SEX (CONSULTA 1 Y 3)

```
--tercera tabla
DROP TABLE IF EXISTS YEAR_MONTH_DAY_TIME_AREA_SUB_AREA_CRIME_AGE_SEX;
CREATE TABLE YEAR_MONTH_DAY_TIME_AREA_SUB_AREA_CRIME_AGE_SEX (
   year_occ INT,
   month_occ INT,
   day_occ INT,
   time_occ TEXT,
   area_name TEXT,
   sub_area TEXT,
   crime_description TEXT,
   vict_age INT,
   vict_sex TEXT,
```

```
   total_count INT,

   PRIMARY KEY  ((year_occ), month_occ, day_occ, area_name, time_occ, sub_area,
crime_description, vict_age, vict_sex , total_count)

);

COPY YEAR_MONTH_DAY_TIME_AREA_SUB_AREA_CRIME_AGE_SEX (year_occ, month_occ,
day_occ, time_occ, area_name, sub_area, crime_description, vict_age, vict_sex, total_count)

FROM '/home/uoclabs/data/YEAR_MONTH_DAY_TIME_AREA_SUB_AREA_CRIME_AGE_SEX.csv'

WITH DELIMITER=',' AND HEADER=TRUE;
```

Using 1 child processes

Starting copy of crimes.year_month_day_time_area_sub_area_crime_age_sex with columns [year_occ, month_occ, day_occ, time_occ, area_name, sub_area, crime_description, vict_age, vict_sex, total_count].
Processed: 895615 rows; Rate: 2339 rows/s; Avg. rate: 5466 rows/s
895615 rows imported from 1 files in 0 day, 0 hour, 2 minutes, and 43.845 seconds (0 skipped).
cqlsh:crimes>

--el csv tiene con la cabecera 895616 lineas, parece correcto

1.2. Consulta 1 (22,5%) Suma de todos los crímenes del área con nombre "Hollywood". La consulta debe retornar dos columnas, el nombre del área que será "Hollywood" y la suma total.

```
cqlsh:crimes> SELECT area_name, SUM(total_count) AS total_crimes FROM
YEAR_MONTH_DAY_TIME_AREA_SUB_AREA_CRIME_AGE_SEX  WHERE
area_name = 'Hollywood';

 area_name | total_crimes
-----------+--------------
 Hollywood |        47508
```

(1 rows)

cqlsh:crimes>

Al igual que SQL, ya que la sintaxis es muy parecida en cql, usamos un select, la columna, el sumatorio en la funcion SUM(), de donde (FROM) y el filtro (WHERE)

1.3 Consulta 2 (22,5%) El Ayuntamiento de la ciudad de Los Ángeles está preocupado por la evolución de los crímenes con armas de asalto en los últimos años. Se pide una consulta que muestre los crímenes con una descripción del arma utilizada 'ASSAULT WEAPON/UZI/AK47/ETC' agrupados por año. Las columnas de la consulta son WEAPON_DESC, YEAR_OCC y la suma de todos los crímenes.

```
cqlsh:crimes> SELECT year_occ, weapon_desc, SUM(total_count) AS total_crimes
FROM YEAR_MONTH_DAY_TIME_AREA_SUB_AREA_WEAPON WHERE
weapon_desc = 'ASSAULT WEAPON/UZI/AK47/ETC' GROUP BY year_occ;
```

```
 year_occ | weapon_desc                | total_crimes
----------+----------------------------+--------------
     2020 | ASSAULT WEAPON/UZI/AK47/ETC |           26
     2021 | ASSAULT WEAPON/UZI/AK47/ETC |           12
     2022 | ASSAULT WEAPON/UZI/AK47/ETC |           20
     2023 | ASSAULT WEAPON/UZI/AK47/ETC |           16

(4 rows)
cqlsh:crimes>
```

Como he creado correctamente la primary key con el orden correcto, puedo agrupar bien por year. He usado simplemente un WHERE, y GROUP BY, que al estar bien definidas las tablas, ha mostrado correctamente la informacion.

1.4 Consulta 3 (22,5%) Se dice que la nochebuena y la noche de fin de año son unas de las noches con más criminalidad del año. Se quiere comparar el total de crímenes del día uno de enero de 2023 y del uno de marzo del mismo año. Sugerencia: el operador OR no se puede utilizar para esta consulta, deberás utilizar el operador IN.

```
cqlsh:crimes> SELECT year_occ, month_occ, day_occ, SUM(total_count) AS
total_crimes  FROM
YEAR_MONTH_DAY_TIME_AREA_SUB_AREA_CRIME_AGE_SEX  WHERE
year_occ = 2023 AND month_occ IN (1, 3) AND day_occ = 1  GROUP BY month_occ,
day_occ;

 year_occ | month_occ | day_occ | total_crimes
----------+-----------+---------+--------------
     2023 |         1 |       1 |         1106
     2023 |         3 |       1 |          833

(2 rows)
cqlsh:crimes>
```

Una vez definida la clave de particion year en la definicion de la tabla, significa que todas estaran en la misma particion, y luego para clustering he usado month, day y el resto de columnas para evitar perder columnas por repetirse la clave unica.

1.5 Consulta 4 (22,5%) Se quiere hacer una comparación entre los años 2022 y 2023 de aquellos crímenes que han usado como arma "STRONG-ARM (HANDS, FIST , FEET OR BODILY FORCE)". La consulta debe retornar las columnas WEAPON_DESC, YEAR_OCC y la suma de todos los crímenes agrupados por esta arma y años.

```
cqlsh:crimes> SELECT weapon_desc, year_occ, SUM(total_count) AS total_crimes
FROM YEAR_MONTH_DAY_TIME_AREA_SUB_AREA_WEAPON WHERE
weapon_desc = 'STRONG-ARM (HANDS, FIST, FEET OR BODILY FORCE)'   AND
year_occ IN (2022, 2023) GROUP BY year_occ;

 weapon_desc                                | year_occ | total_crimes
--------------------------------------------+----------+-------------
 STRONG-ARM (HANDS, FIST, FEET OR BODILY FORCE) |     2022 |        41888
 STRONG-ARM (HANDS, FIST, FEET OR BODILY FORCE) |     2023 |        41661

(2 rows)
cqlsh:crimes>
```

Agrupo por year, y el resto, es simplemente las columnas que quiero, el sumatorio, con un alias.

1.6 Warning en consultas (5%) Algunas de las consultas ejecutadas anteriormente retornan la advertencia *warning:Aggregation query used without partition key*. ¿Qué significa?

Indica que la consulta (de agregacion del estilo SUM, COUNT, etc) se hizo sin usar una clave definida como de particion. En mi caso lo use y no he tenido ese problema.

Ejercicio 2. Neo4j

Primeros pasos

```
uoclabs@ML-RefVm-731264:~$ cypher-shell
username: neo4j
password: ****
Connected to Neo4j 4.1.0 at neo4j://localhost:7687 as user neo4j.
Type :help for a list of available commands or :exit to exit the shell.
Note that Cypher queries must end with a semicolon.
neo4j@neo4j>
neo4j@neo4j> :USE neo4j
neo4j@neo4j> MATCH (n) DETACH DELETE n;
0 rows available after 381 ms, consumed after another 0 ms
neo4j@neo4j> SHOW INDEXES;
Invalid input 'I': expected whitespace, comment, DATABASE, DATABASES, DEFAULT, POPULATED, ALL, ROLES, USERS or show privilege scope (line 2, column 6 (offset: 6))
"SHOW INDEXES;"
     ^
neo4j@neo4j> CALL db.indexes();
+-----------------------------------------------------------------------------------------------------------------+
| id | name | state | populationPercent | uniqueness | type | entityType | labelsOrTypes | properties | provider |
+-----------------------------------------------------------------------------------------------------------------+
+-----------------------------------------------------------------------------------------------------------------+

0 rows available after 140 ms, consumed after another 27 ms
neo4j@neo4j>
Interrupted (Note that Cypher queries must end with a semicolon. Type :exit to exit the shell.)
neo4j@neo4j> :exit

Bye!
```

Modificación de CreateDB2cypher

Da errores de importación, probablemente por la versión de neo4j

Unknown function 'toFloatOrNull' (line 29, column 20 (offset: 879))

"SET c.gender = CASE row.gender WHEN "None" THEN NULL ELSE row.gender END, c.height = toFloatOrNull(row.height), c.weight = toFloatOrNull(row.weight), c.born = toIntegerOrNull(row.year_born), c.died = toIntegerOrNull(row.year_died), c.descripcion = row.description"

Por tanto tengo que sustituir las lineas con CASE y COALESCE:

CREATE CONSTRAINT ON (p:Species) ASSERT p.id IS UNIQUE;

CREATE CONSTRAINT ON (p:Character) ASSERT p.id IS UNIQUE;

CREATE CONSTRAINT ON (p:Film) ASSERT p.id IS UNIQUE;

```
CREATE CONSTRAINT ON (p:Planet) ASSERT p.id IS UNIQUE;

CREATE CONSTRAINT ON (p:Organization) ASSERT p.id IS UNIQUE;

CREATE CONSTRAINT ON (p:Affiliation) ASSERT p.id IS UNIQUE;


LOAD CSV WITH HEADERS FROM "https://raw.githubusercontent.com/planetacomputer/neo4j-
starwars/main/dataset/films.csv"
  AS row
  UNWIND split(row.producer, ",") AS producer
  MERGE (f:Film {name: trim(row.title),
    opening: row.opening_crawl})
  MERGE (d:Person {name: trim(row.director)})
  MERGE (f)-[:DIRECTED_BY]->(d)
  MERGE (p:Person {name: trim(producer)})
  MERGE (p)<-[:PRODUCED_BY]-(f);


LOAD CSV WITH HEADERS FROM "https://raw.githubusercontent.com/planetacomputer/neo4j-
starwars/main/dataset/characters.csv"
  AS row
  MERGE (c:Character {name: row.name})
  FOREACH(
    it IN
      CASE row.homeworld WHEN "None"
        THEN null
        WHEN "Unknown" THEN null
        ELSE trim(row.homeworld)
      END |
      MERGE (p:Planet {name: it})
      MERGE (c)-[:IS_HOMEWORLD]->(p) )
```

```
  FOREACH(
    it IN
      CASE row.species WHEN "Unknown"
        THEN null
        ELSE trim(row.species)
      END |
      MERGE (s:Species {name: it})
      MERGE (c)-[:IS_OF_SPECIE]->(s)
      )
  SET
    c.gender = CASE row.gender WHEN "None"
        THEN null
        ELSE row.gender
      END,
    c.height = CASE WHEN row.height IS NULL OR row.height = "None" THEN NULL ELSE
toFloat(row.height) END,
    c.weight = CASE WHEN row.weight IS NULL OR row.weight = "None" THEN NULL ELSE
toFloat(row.weight) END,
    c.born = CASE WHEN row.year_born IS NULL OR row.year_born = "None" THEN NULL ELSE
toInteger(row.year_born) END,
    c.died = CASE WHEN row.year_died IS NULL OR row.year_died = "None" THEN NULL ELSE
toInteger(row.year_died) END,
    c.descripcion = row.description;

LOAD CSV WITH HEADERS FROM "https://raw.githubusercontent.com/planetacomputer/neo4j-
starwars/main/dataset/planets.csv"
  AS row
  UNWIND split(row.residents, ",") AS residents
  UNWIND split(row.films, ",") AS film
  MERGE (p:Planet {name: trim(row.name)})
```

```
MERGE (f:Film {name: trim(film)})

MERGE (p)-[:APPEARS_IN]->(f)

SET

    p.diameter = CASE WHEN row.diameter IS NULL OR row.diameter = "None" THEN NULL ELSE
toInteger(row.diameter) END,

    p.rotation_period = CASE WHEN row.rotation_period IS NULL OR row.rotation_period =
"None" THEN NULL ELSE toInteger(row.rotation_period) END,

    p.orbital_period = CASE WHEN row.orbital_period IS NULL OR row.orbital_period = "None"
THEN NULL ELSE toInteger(row.orbital_period) END,

    p.gravity = CASE WHEN row.gravity IS NULL OR row.gravity = "None" THEN NULL ELSE
toFloat(replace(row.gravity, "standard", "")) END,

    p.population = CASE WHEN row.population IS NULL OR row.population = "None" THEN NULL
ELSE toFloat(row.population) END,

    p.surface_water = CASE WHEN row.surface_water IS NULL OR row.surface_water = "None"
THEN NULL ELSE toInteger(row.surface_water) END;


LOAD CSV WITH HEADERS FROM "https://raw.githubusercontent.com/planetacomputer/neo4j-
starwars/main/dataset/organizations.csv"

AS row

UNWIND split(row.leader, ",") AS leader

UNWIND split(row.members, ",") AS member

UNWIND split(row.films, ",") AS film

MERGE (o:Organization {name: row.name})

MERGE (c:Character {name: trim(member)})

MERGE (f:Film {name: trim(film)})

MERGE (o)<-[:LEADER_OF]-(c)

MERGE (o)-[:APPEARS_IN]->(f)

FOREACH(

    it IN

        CASE trim(row.affiliation)
```

```
            WHEN "None"
            THEN null
            ELSE trim(row.affiliation)
        END |
        MERGE (a:Affiliation {name: it})
        MERGE (o)-[:BELONGS_TO]->(a)
        )
  SET
    o.founded = CASE WHEN row.founded IS NULL OR row.founded = "None" THEN NULL ELSE
toInteger(row.founded) END,
    o.dissolved = CASE WHEN row.dissolved IS NULL OR row.dissolved = "None" THEN NULL ELSE
toInteger(row.dissolved) END,
    o.description = row.description;

LOAD CSV WITH HEADERS FROM "https://raw.githubusercontent.com/planetacomputer/neo4j-
starwars/main/dataset/species.csv"
  AS row
  MERGE (s:Species {name: row.name})
  FOREACH(
    it IN
      CASE trim(replace(row.classification, "Unknown",""))
        WHEN "" THEN null
        ELSE trim(row.classification)
      END |
      MERGE (g:Classification {name: it})
      MERGE (s)-[:BELONGS_TO]->(g) )
  FOREACH(
    it IN
      CASE trim(replace(replace(row.homeworld, "Unknown",""), "Various", "" ))
```

```
            WHEN "" THEN null
            ELSE trim(row.homeworld)
        END |
        MERGE (p:Planet {name: it})
        MERGE (s)-[:IS_HOMEWORLD]->(p) )
  SET
      s.designation = row.designation,
      s.average_height = CASE WHEN row.average_height IS NULL OR row.average_height = "None" THEN NULL ELSE toFloat(row.average_height) END,
      s.average_lifespan = CASE WHEN row.average_lifespan IS NULL OR row.average_lifespan = "None" THEN NULL ELSE toInteger(row.average_lifespan) END,
      s.language = row.language;
```

Carga

```
uoclabs@ML-RefVm-731264:~$ cat CreateDB2.cypher | cypher-shell -u neo4j -p 1234
uoclabs@ML-RefVm-731264:~$ cypher-shell
username: neo4j
password: ****
Connected to Neo4j 4.1.0 at neo4j://localhost:7687 as user neo4j.
Type :help for a list of available commands or :exit to exit the shell.
Note that Cypher queries must end with a semicolon.
neo4j@neo4j> MATCH (n) RETURN COUNT(n);
+----------+
| COUNT(n) |
+----------+
| 240      |
+----------+

1 row available after 27 ms, consumed after another 1 ms
neo4j@neo4j> MATCH ()-[r]->() RETURN COUNT(r);
```

```
+----------+
| COUNT(r) |
+----------+
| 346      |
+----------+
```

1 row available after 21 ms, consumed after another 0 ms

neo4j@neo4j>

Consulta 1 (10%) Mostrar por pantalla el recuento de personas que han producido la película titulada "Return of the Jedi" mostrando, no sólo el número de personas, sino también una lista con sus nombres.

```
neo4j@neo4j> MATCH (f:Film)-[:PRODUCED_BY]->(p:Person)
             WHERE f.name = "Return of the Jedi"
             RETURN COUNT(p) AS num_producers, COLLECT(p.name) AS producers;
+-------------------------------------------------------------------------+
| num_producers | producers                                               |
+-------------------------------------------------------------------------+
| 3             | ["Rick McCallum", "George Lucas", "Howard G. Kazanjian"] |
+-------------------------------------------------------------------------+
```

1 row available after 82 ms, consumed after another 1 ms

Como podemos ver, tenemos MATCH para buscar patrones, luego el nodo Film , la relacion entre los nodos produced_by, y el nodo Person.

Aquí busco las películas y sus productores usando una relación PRODUCED_BY, luego agrupo los nombres con COLLECT para obtener una lista.

Consulta 2 (10%) Supón que para dar respuesta a la consulta 1, uno de nuestros colegas nos propone realizar la siguiente consulta:

MATCH (p:Person)<--(f:Film)

WHERE f.name = "Return of the Jedi"

RETURN …

¿Es una solución válida? Justifica el porqué utilizando su ejecución para validar vuestra argumentación.

Si yo uso:

```
neo4j@neo4j> MATCH (p:Person)<--(f:Film)
             WHERE f.name = "Return of the Jedi"
             RETURN COUNT(p) AS count, COLLECT(p.name) AS names;
+-------------------------------------------------------------------------------+
| count | names                                                                 |
+-------------------------------------------------------------------------------+
| 4     | ["Rick McCallum", "Richard Marquand", "George Lucas", "Howard G. Kazanjian"] |
```

```
+--------------------------------------------------------------------------------+
```

1 row available after 6 ms, consumed after another 2 ms

Me devuelve 4 nombres, el cuarto es cualquier persona que conecte con Film desde Person. Quizas la persona que sale de mas, Richard Marquand, tiene otro puesto en la película:

```
neo4j@neo4j> MATCH (f:Film)-[r]-(p:Person)
             WHERE f.name = "Return of the Jedi"
             RETURN p.name AS person, TYPE(r) AS relationship;
+-------------------------------------+
| person               | relationship  |
+-------------------------------------+
| "Rick McCallum"      | "PRODUCED_BY" |
| "Richard Marquand"   | "DIRECTED_BY" |
| "George Lucas"       | "PRODUCED_BY" |
| "Howard G. Kazanjian" | "PRODUCED_BY" |
+-------------------------------------+

4 rows available after 107 ms, consumed after another 0 ms
neo4j@neo4j>
```

En esta consulta lo unico que he usado diferente es devolver TYPE(r) que devuelve el tipo de relacion.

Por tanto, esta consulta no es válida porque incluye a cualquier persona conectada con la película, sin filtrar por la relación PRODUCED_BY

Consulta 3 (15%) Presentar las películas donde aparezcan más de 2 planetas, indicando el nombre de la película, el número de planetas y el nombre de los mismos. El listado deberá estar ordenado descendentemente por el número de planetas que aparecen en la película y, en caso de tener el mismo número por el nombre de las películas.

```
neo4j@neo4j> MATCH (p:Planet)-[:APPEARS_IN]->(f:Film)
             WITH f, COUNT(p) AS planet_count, apoc.coll.sort(COLLECT(p.name)) AS planet_names
```

```
        WHERE planet_count > 2
        RETURN f.name AS film, planet_count, planet_names
        ORDER BY planet_count DESC, planet_names ASC, f.name ASC;
+-----------------------------------------------------------------------------------------------+
| film                  | planet_count | planet_names                                           |
+-----------------------------------------------------------------------------------------------+
| "Revenge of the Sith"   | 5            | ["Alderaan", "Coruscant", "Kashyyyk", "Naboo", "Tatooine"] |
| "Attack of the Clones"  | 5            | ["Coruscant", "Geonosis", "Kamino", "Naboo", "Tatooine"]   |
| "A New Hope"            | 3            | ["Alderaan", "Tatooine", "Yavin 4"]                        |
| "The Clone Wars"        | 3            | ["Coruscant", "Geonosis", "Rodia"]                         |
| "The Rise of Skywalker" | 3            | ["Coruscant", "Jakku", "Yavin 4"]                          |
| "The Phantom Menace"    | 3            | ["Coruscant", "Naboo", "Tatooine"]                         |
+-----------------------------------------------------------------------------------------------+

6 rows available after 177 ms, consumed after another 11 ms
neo4j@neo4j>
```

Uso apoc.coll.sort para ordenar alfabéticamente los nombres de los planetas antes de mostrarlos. (*) .Luego ordeno por cantidad de planetas, luego alfabéticamente por nombres de planetas y títulos de películas para mantener el orden del enunciado.

Consulta 4 (20%) Mostrar los apoyos recibidos por los diferentes gobiernos (Affiliation). Para cada gobierno, indicad

su nombre, la lista de los nombres de las organizaciones que le dan apoyo (sin repetidos) y una lista con los nombres de los líderes de esas organizaciones. Ordenad los resultados de forma descendente, en función del número de líderes que han apoyado a cada organización.

```
neo4j@neo4j> MATCH (o:Organization)-[:BELONGS_TO]->(a:Affiliation)OPTIONAL MATCH (c:Character)-[:LEADER_OF]->(o)
             RETURN a.name AS affiliation,COLLECT(DISTINCT o.name) AS organizations, COLLECT(DISTINCT c.name) AS leaders
             ORDER BY SIZE(leaders) DESC;
+-----------------------------------------------------------------------------------------------------------------------------+
| affiliation         | organizations                  | leaders                                                      |
+-----------------------------------------------------------------------------------------------------------------------------+
| "New Republic"      | ["Rebel Alliance", "Resistance"] | ["Han Solo", "Luke Skywalker", "Lando Calrissian", "Finn", "Rey", "Poe Dameron"] |
| "Galactic Republic" | ["Jedi Order"]                 | ["Obi-Wan Kenobi", "Luke Skywalker", "Anakin Skywalker"]     |
| "Galactic Empire"   | ["Sith Order"]                 | ["Darth Maul", "Darth Tyranus", "Darth Vader"]               |
| "Jedi Order"        | ["Galactic Republic"]          | ["Bail Organa", "Padmé Amidala"]                             |
| "Separatists"       | ["Trade Federation"]           | ["Lott Dod", "Rune Haako"]                                   |
+-----------------------------------------------------------------------------------------------------------------------------+

5 rows available after 24 ms, consumed after another 8 ms
neo4j@neo4j>
```

Esta consulta busca todas las organizaciones relacionadas con afiliaciones con la relacion BELONGS_TO

Busca ademas los personajes que lideran esas organizaciones, toma los nombres distintos en una lista, y lo ordena por tamano.

Consulta 5 (20%) Presentar el nombre de los gobiernos (Affiliation) a los cuales pertenecen organizaciones lideradas por personajes de la familia Skywalker. Observad que una organización puede haber tenido distintos líderes en diferentes momentos. Para esta consulta interesa obtener, en una columna el nombre del gobierno y, en una segunda columna, el listado de los líderes de las organizaciones lideradas por algún (o alguna) Skywalker.

```
neo4j@neo4j> MATCH (c:Character)-[:LEADER_OF]->(o:Organization)-
[:BELONGS_TO]->(a:Affiliation)WHERE c.name CONTAINS "Skywalker"RETURN
a.name AS government, COLLECT(DISTINCT c.name) AS leaders;
+-----------------------------------------------------------+
| government          | leaders                             |
+-----------------------------------------------------------+
| "Galactic Republic" | ["Luke Skywalker", "Anakin Skywalker"] |
| "New Republic"      | ["Luke Skywalker"]                  |
+-----------------------------------------------------------+

2 rows available after 28 ms, consumed after another 7 ms
neo4j@neo4j>
```

Aqui identifico a gobiernos (afiliacion) que pertenecen a organizaciones cuyo lider (personaje) contiene Skywalker. Por ultimo uso COLLECT para que devuelva una lista y DISTINCT para que no repitan nombres.

Aqui vemos que la consulta busco el personaje que sea lider de una organizacion, que pertenezca a una afiliacion, luego en el where que contenga Skywalker y devuelve el gobierno, y una lista de lideres.

Consulta 6 (25%) Con el objetivo de evaluar la variedad de organizaciones que aparecen en cada capítulo de la saga y, por tanto, cómo de complicada puede llegar a ser la trama, se desea investigar sobre el número de organizaciones que aparecen en las diferentes películas. Para ello se solicita un listado de todas las organizaciones, donde aparezca por cada organización: el nombre de la organización, el número de películas donde aparece, y otra columna denominada 'apariciones' en la que se indique: 'Baja' si aparecen en menos de 4 películas, 'Media' si aparecen en 4 o 5 películas y 'Alta' si aparecen en 6 o más películas. Se desea que las organizaciones se muestren ordenadas por su fecha de fundación y, si hay alguna repetición en la fecha de fundación, que se muestren en orden alfabético ascendente por el título de la organización.

```
neo4j@neo4j> MATCH (o:Organization)-[:APPEARS_IN]->(f:Film)
                    WITH o, COUNT(f) AS n_peliculas, COLLECT(f.name) AS films
                    WITH o, n_peliculas,
                        CASE WHEN n_peliculas < 4 THEN 'Baja'WHEN
n_peliculas <= 5 THEN 'Media'ELSE 'Alta'
                        END AS apariciones
                    RETURN o.name AS organizacion, n_peliculas, apariciones //,
o.founded
                    ORDER BY o.founded ASC, o.name ASC;
+-----------------------------------------------+
| organizacion        | n_peliculas | apariciones |
+-----------------------------------------------+
| "Galactic Republic" | 3           | "Baja"      |
```

"Jedi Order"	6	"Alta"
"Sith Order"	6	"Alta"
"Trade Federation"	3	"Baja"
"Rebel Alliance"	3	"Baja"
"Galactic Empire"	4	"Media"
"First Order"	3	"Baja"
"Resistance"	3	"Baja"

+--+

8 rows available after 48 ms, consumed after another 6 ms

neo4j@neo4j>

Clasifico el número de películas en categorías ('Baja', 'Media', 'Alta') luego ordeno las organizaciones por fecha de fundación y, en caso de empate, alfabéticamente por su nombre.

Ejercicio 3. MongoDB

Carga de datos

```
uoclabs@ML-RefVm-731264:~$ git clone https://github.com/uoc-bbdd-no-tradicionales/mongodb-sample-dataset.git && cd mongodb-sample-dataset && ./script.sh
Cloning into 'mongodb-sample-dataset'...
remote: Enumerating objects: 9, done.
remote: Counting objects: 100% (9/9), done.
remote: Compressing objects: 100% (9/9), done.
remote: Total 9 (delta 0), reused 9 (delta 0), pack-reused 0 (from 0)
Receiving objects: 100% (9/9), 4.64 MiB | 9.93 MiB/s, done.
sample_analytics
2024-12-30T06:58:57.240+0000    connected to: mongodb://localhost:27017/
2024-12-30T06:58:57.240+0000    dropping: sample_analytics.accounts
2024-12-30T06:58:57.665+0000    1746 document(s) imported successfully. 0 document(s) failed to import.
2024-12-30T06:58:57.693+0000    connected to: mongodb://localhost:27017/
2024-12-30T06:58:57.693+0000    dropping: sample_analytics.customers
2024-12-30T06:58:57.852+0000    500 document(s) imported successfully. 0 document(s) failed to import.
2024-12-30T06:58:57.876+0000    connected to: mongodb://localhost:27017/
2024-12-30T06:58:57.877+0000    dropping: sample_analytics.transactions
2024-12-30T06:59:00.483+0000    1746 document(s) imported successfully. 0 document(s) failed to import.
uoclabs@ML-RefVm-731264:~/mongodb-sample-dataset$
```

Use y show

```
uoclabs@ML-RefVm-731264:~/mongodb-sample-dataset$ mongo
MongoDB shell version v4.4.20
connecting to: mongodb://127.0.0.1:27017/?compressors=disabled&gssapiServiceName=mongodb
Implicit session: session { "id" : UUID("48b1c8ba-7684-4875-80c8-cb25333b5b26") }
MongoDB server version: 4.4.20
---
The server generated these startup warnings when booting:
        2024-12-30T02:01:26.059+00:00: Using the XFS filesystem is strongly recommended with the WiredTiger storage engine. See
http://dochub.mongodb.org/core/prodnotes-filesystem
        2024-12-30T02:01:34.755+00:00: Access control is not enabled for the database. Read and write access to data and configuration is
unrestricted
        2024-12-30T02:01:34.755+00:00: /sys/kernel/mm/transparent_hugepage/enabled is 'always'. We suggest setting it to 'never'
---
---
        Enable MongoDB's free cloud-based monitoring service, which will then receive and display
        metrics about your deployment (disk utilization, CPU, operation statistics, etc).

        The monitoring data will be available on a MongoDB website with a unique URL accessible to you
        and anyone you share the URL with. MongoDB may use this information to make product
        improvements and to suggest MongoDB products and deployment options to you.

        To enable free monitoring, run the following command: db.enableFreeMonitoring()
        To permanently disable this reminder, run the following command: db.disableFreeMonitoring()
---
> use sample_analytics
switched to db sample_analytics
> show collections
accounts
customers
transactions
>
```

3.1 Consulta 1 (10%) Se necesita saber el correo electrónico (email) junto con detalles de nivel de suscripción (tier_and_details) del cliente con nombre de usuario (username) “valenciajennifer”. La consulta deberá mostrar el correo electrónico, las cuentas del cliente y todos los detalles del nivel de suscripción. El resultado de la consulta no debe mostrar ningún campo más.

```
> db.customers.find({ username: "valenciajennifer" },{ email: 1, accounts: 1,
"tier_and_details": 1, _id: 0 }).pretty();
{
        "email" : "cooperalexis@hotmail.com",
        "accounts" : [
                116508
        ],
        "tier_and_details" : {
                "c06d340a4bad42c59e3b6665571d2907" : {
                        "tier" : "Platinum",
                        "benefits" : [
                                "dedicated account representative"
                        ],
                        "active" : true,
                        "id" : "c06d340a4bad42c59e3b6665571d2907"
                },
                "5d6a79083c26402bbef823a55d2f4208" : {
                        "tier" : "Bronze",
                        "benefits" : [
                                "car rental insurance",
                                "concierge services"
                        ],
                        "active" : true,
                        "id" : "5d6a79083c26402bbef823a55d2f4208"
                },
                "b754ec2d455143bcb0f0d7bd46de6e06" : {
                        "tier" : "Gold",
                        "benefits" : [
                                "airline lounge access"
                        ],
                        "active" : true,
                        "id" : "b754ec2d455143bcb0f0d7bd46de6e06"
```

```
            }
        }
}
>
```

Filtro por el usuario específico y uso la clave valor para mostrar o no (1 o 0), para limitar los campos mostrados.

3.2 Consulta 2 (15%) Se quiere hacer un ranking de los productos con más éxito entre los clientes. Se pide un recuento de todos los productos de las cuentas ordenados de manera descendente.

```
>
> db.accounts.aggregate([
...   { $unwind: "$products" },
...   { $group: { _id: { product: "$products" }, count: { $sum: 1 } } },
...   { $sort: { count: -1 } }
... ]);
{ "_id" : { "product" : "InvestmentStock" }, "count" : 1746 }
{ "_id" : { "product" : "CurrencyService" }, "count" : 742 }
{ "_id" : { "product" : "Brokerage" }, "count" : 741 }
{ "_id" : { "product" : "InvestmentFund" }, "count" : 728 }
{ "_id" : { "product" : "Commodity" }, "count" : 720 }
{ "_id" : { "product" : "Derivatives" }, "count" : 706 }
>
```

Separo cada producto en una fila distinta para poder contar correctamente su aparición y he agrupado por producto y cuento cuántas veces aparece en todas las cuentas.

3.3 Consulta 3 (25%) Las diez transacciones de menor importe del cliente con nombre de usuario (username) 'timothyvelasquez' . El listado de transacciones debe retornar el identificador de la cuenta account_id y el importe (amount).

```
> db.customers.find(
...   { username: "timothyvelasquez" },
...   { accounts: 1, _id: 0 }
... );
{ "accounts" : [ 972116, 755845, 510435, 206973 ] }
>


>
> db.transactions.find(
...   { account_id: { $in: [972116, 755845, 510435, 206973] } },
...   { account_id: 1, amount: 1, _id: 0 }
... ).sort({ amount: 1 }).limit(10);
{ "account_id" : 206973 }
{ "account_id" : 510435 }
{ "account_id" : 972116 }
{ "account_id" : 755845 }
>
```

Uso $in para filtrar todas las transacciones de las cuentas del cliente, buscando solo las más pequeñas con sort y limit.

Si bien no he entendio bien si puedo copiar o no, ya que dice el enunciado de 2 consultas, y no se me ha ocurrido otra forma que asi:

```
>
> var accounts = db.customers.findOne(
...   { username: "timothyvelasquez" },
...   { accounts: 1, _id: 0 }
... ).accounts;
>
> db.transactions.find(
...   { account_id: { $in: accounts } },
...   { account_id: 1, amount: 1, _id: 0 }
... ).sort({ amount: 1 }).limit(10);
{ "account_id" : 206973 }
{ "account_id" : 510435 }
{ "account_id" : 972116 }
{ "account_id" : 755845 }
>
>
```

Primero se obtiene las cuentas del cliente y las uso para filtrar en una consulta separada. Es otra opcion para reutilizar las cuentas

3.4 Consulta 4 (25%) Se quiere dar de alta una nueva cuenta para el cliente con nombre de usuario (username)'timothyayers'. Los datos de la cuenta son:{"account_id": 872399,"limit": 10000,"products": ["InvestmentStock"]}

```
>
>
> db.accounts.insertOne({
...   account_id: 872399,
```

```
...   limit: 10000,
...   products: ["InvestmentStock"]
... });
{
        "acknowledged" : true,
        "insertedId" : ObjectId("67724a9fd47e44b167e2b41e")
}
>
```

Compruebo si esta insertado bien:

```
>
> db.accounts.find({ account_id: 872399 }).pretty();
{
        "_id" : ObjectId("67724a9fd47e44b167e2b41e"),
        "account_id" : 872399,
        "limit" : 10000,
        "products" : [
                "InvestmentStock"
        ]
}
>
```

3.5 Consulta 5 (25%) En las cuentas relacionadas con carteras de valores, suele ser extraño que haya cuentas compartidas (como pasaría con las cuentas corrientes). Se quieren conocer las cuentas que pertenecen a más de un cliente. Si las hay, también se quiere saber quiénes son los propietarios. De los propietarios nos interesa saber el nombre, la dirección y los números de sus cuentas. La consulta con el resultado final no debe mostrar ningún campo más.

```
> db.customers.aggregate([
...   { $unwind: "$accounts" },
...   { $group: { _id: "$accounts", count: { $sum: 1 }, owners: { $push: "$name" } } },
...   { $match: { count: { $gt: 1 } } }
... ]);
{ "_id" : 627788, "count" : 2, "owners" : [ "Ashley Rodriguez", "Shawn Austin" ] }
>
```

Con $unwind separo las cuentas en filas individuales para analizarlas por separado, con $group agrupo por cada cuenta, cuento cuantos clientes la comparten, y guardo sus nombres en una lista. Por ultimo con $match, filtro para mostrar solo las cuentas compartidas por más de un cliente

```
> db.customers.find(
... { accounts: { $in: [627788] } },
... { name: 1, address: 1, accounts: 1, _id: 0 }).pretty();
{
        "name" : "Ashley Rodriguez",
        "address" : "94038 Luis Garden\nWilliamsstad, MI 51943",
        "accounts" : [
```

```
                249078,
                660047,
                627788,
                428217,
                526519,
                814901
        ]
}
{
        "name" : "Shawn Austin",
        "address" : "84228 Alison Rest Suite 507\nTimothyshire, NC 75240",
        "accounts" : [
                693557,
                73934,
                627788,
                539248,
                390126,
                533671
        ]
}
```

Aqui simplemente usando $in, busco por ese numero de cuenta.

Si no se pudiera hacer asi, como la consulta 3, podemos hacerlo con variables:

```
>
> var accountIds = db.customers.aggregate([
...   { $unwind: "$accounts" },
...   { $group: { _id: "$accounts", count: { $sum: 1 } } },
...   { $match: { count: { $gt: 1 } } }
... ]).map(a => a._id);
>
```

```
> db.customers.find(
...   { accounts: { $in: accountIds } },
...   { name: 1, address: 1, accounts: 1, _id: 0 }
... ).pretty();
{
        "name" : "Ashley Rodriguez",
        "address" : "94038 Luis Garden\nWilliamsstad, MI 51943",
        "accounts" : [
                249078,
                660047,
                627788,
                428217,
                526519,
                814901
        ]
}
{
        "name" : "Shawn Austin",
        "address" : "84228 Alison Rest Suite 507\nTimothyshire, NC 75240",
        "accounts" : [
                693557,
                73934,
                627788,
                539248,
                390126,
                533671
        ]
}
>
```

En var (….) uso la agregacion para identificar cuentas compartidas, primero las descompongo con unwind, para analizarlas por separado, luego las agrupo por cada cuenta, y filtro con match para quedarme solamente con cuentas con mas de 1 count

En db.customers.find (…) Busco los clientes que tienen cuentas compartidas identificadas en la primera consulta como accountIds, uso pretty() para que sea legible facilmente.

Bibliografia y fuentes consultadas

(*)https://neo4j.com/docs/apoc/current/overview/apoc.coll/apoc.coll.sort/

En general he usado la documentacion oficial y el aula virtual:

https://cassandra.apache.org/doc/stable/cassandra/cql/dml.html

neo4j.com

https://www.mongodb.com/docs/manual/