Ús de bases de dades NoSQL
PRA1

Marc Bracons Cucó

07/06/2024

Exercici 1: Cassandra

Càrrega de dades

Hem iniciat Cassandra i hem seguit els passos descrits a l'enunciat de la pràctica

```
Connected to Test Cluster at 127.0.0.1:9042.
[cqlsh 5.0.1 | Cassandra 3.11.11 | CQL spec 3.4.4 | Native protocol v4]
Use HELP for help.
cqlsh> CREATE KEYSPACE CRIMES WITH REPLICATION ={ 'class':
'SimpleStrategy','replication_factor': 1};
cqlsh> USE CRIMES;
cqlsh:crimes> CREATE TABLE TESTING_AGGREGATE
          ... (
          ... COLUMNA1 TEXT,
          ... COLUMNA2 TEXT,
          ... COLUMNA3 TEXT,
          ... COLUMNA4 TEXT,
          ... PRIMARY KEY (COLUMNA1,COLUMNA2,COLUMNA3)
          ... );
cqlsh:crimes> COPY CRIMES.TESTING_AGGREGATE (COLUMNA1, COLUMNA2, COLUMNA3,
COLUMNA4) FROM
          ... '/home/student/Escritorio/Cassandra/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:       4 rows/s; Avg. rate:       5 rows/s
2 rows imported from 1 files in 0.368 seconds (0 skipped).
cqlsh:crimes> SELECT *
          ... 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>
```

Exercici, modificació de dades

```
cqlsh:crimes> 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> UPDATE TESTING_AGGREGATE
          ... SET COLUMNA4 = 'valor4'
          ... WHERE COLUMNA1 = 'clave1_3'  AND COLUMNA2 = 'clave2_3' AND
COLUMNA3 = 'clave3_3';
cqlsh:crimes> SELECT * FROM TESTING_AGGREGATE;

 columna1 | columna2 | columna3 | columna4
----------+----------+----------+----------
 clave1_1 | clave2_1 | clave3_1 |   valor3
 clave1_2 | clave2_2 | clave3_2 |   valor2
 clave1_3 | clave2_3 | clave3_3 |   valor4

(3 rows)
cqlsh:crimes>
```

Amb la comanda INSERT INTO TESTING_AGGREGATE hem inserit una nova fila a la taula **TESTING_AGGREGATE** amb els valors especificats a cada columna.

En canvi, amb UPDATE TESTING_AGGREGATE hem actualitzat el valor de **COLUMNA4** a **valor4** per la fila on **COLUMNA1**, **COLUMNA2** i **COLUMNA3** coincideixen amb **clave1_3**, **clave2_3** i **clave3_3** respectivament. Com aquesta fila no existeix, Cassandra ha creat automàticament una nova fila amb aquests valors i llavors si que hi ha una fila que compleix els requisits i estableix **COLUMNA4** a **valor4**.

Això passa perquè Cassandra tracta l'operació d'actualització com un *upsert* (*insert* o *update*). En una base de dades relacional tradicional, una comanda **UPDATE** només actualitza les files existents, és a dir, si les claus específiques no existeixen, no es crearà una nova fila.

Base de dades per consultes

Carrega de dades:

```
cqlsh:crimes> CREATE TABLE area_sub_area_desc_age_sex (
          ...     area_name TEXT,
          ...     sub_area TEXT,
          ...     crime_description TEXT,
          ...     vict_age INT,
          ...     vict_sex TEXT,
          ...     count INT,
          ...     PRIMARY KEY ((area_name, sub_area), crime_description,
vict_age, vict_sex)
          ... );
cqlsh:crimes> CREATE TABLE year_month_day_time_area_sub_area_crime_age_sex (
          ...     year_occ INT,
          ...     month_occ INT,
          ...     day_occ INT,
          ...     time_occ INT,
          ...     area_name TEXT,
          ...     sub_area TEXT,
          ...     crime_description TEXT,
          ...     vict_age INT,
          ...     vict_sex TEXT,
          ...     total INT,
          ...     PRIMARY KEY ((year_occ, month_occ), day_occ, time_occ,
area_name, sub_area, crime_description, vict_age, vict_sex)
          ... );
cqlsh:crimes> COPY area_sub_area_desc_age_sex (area_name, sub_area,
crime_description, vict_age, vict_sex, count) FROM
'/home/student/Escritorio/Cassandra/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, count].
Processed: 529552 rows; Rate:   10168 rows/s; Avg. rate:   20699 rows/s
529552 rows imported from 1 files in 25.584 seconds (0 skipped).
cqlsh:crimes>
cqlsh:crimes> 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) FROM
'/home/student/Escritorio/Cassandra/YEAR_MONTH_DAY_TIME_AREA_SUB_AREA_CRIME_AG
E_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].
Processed: 895615 rows; Rate:   11753 rows/s; Avg. rate:   20216 rows/s
895615 rows imported from 1 files in 44.303 seconds (0 skipped).
cqlsh:crimes>
```

Consulta 1 (20%)

Total de crims produïts durant els mesos de gener, febrer, març, abril, maig i juny de l'any 2020. La consulta ha de retornar l'any, el mes i el recompte total de crims en aquest període.

Suggerència: l'operador OR no es pot fer servir per aquesta consulta, hauràs de fer servir l'operador IN.

```
cqlsh:crimes> SELECT year_occ, month_occ, SUM(total) AS
total_crimes
          ... FROM
year_month_day_time_area_sub_area_crime_age_sex
          ... WHERE year_occ = 2020 AND month_occ IN (1, 2, 3,
4, 5, 6)
          ... GROUP BY year_occ, month_occ;

 year_occ | month_occ | total_crimes
----------+-----------+--------------
     2020 |         1 |        18524
     2020 |         2 |        17261
     2020 |         3 |        16175
     2020 |         4 |        15689
     2020 |         5 |        17215
     2020 |         6 |        17042

(6 rows)

Warnings :
Aggregation query used on multiple partition keys (IN
restriction)
```

Consulta 2 (20%)

Total d'homicidis. Els homicidis es corresponen amb el total de delictes amb CRIME_DESCRIPTION = 'CRIMINAL HOMICIDE'.

Restricció: No es pot fer servir la clàusula ALLOW FILTERING degut a l'impacte negatiu en el rendiment del clúster i perquè serà retirada en les properes versions de Cassandra.

```
cqlsh:crimes> CREATE INDEX ON
year_month_day_time_area_sub_area_crime_age_sex
(crime_description);
cqlsh:crimes> SELECT sum(total) FROM
year_month_day_time_area_sub_area_crime_age_sex WHERE
crime_description = 'CRIMINAL HOMICIDE';
ReadFailure: Error from server: code=1300 [Replica(s) failed to
execute read] message="Operation failed - received 0 responses
and 1 failures" info={'failures': 1, 'received_responses': 0,
'required_responses': 1, 'consistency': 'ONE'}
cqlsh:crimes> SELECT sum(total) FROM
year_month_day_time_area_sub_area_crime_age_sex WHERE
crime_description = 'CRIMINAL HOMICIDE';

 system.sum(total)
-------------------
              1509

(1 rows)

Warnings :
Aggregation query used without partition key

cqlsh:crimes>
```

Consulta 3 (15%)

Total de delictes amb CRIME_DESCRIPTION = 'PICKPOCKET' (carteristes).

Restricció: No es pot fer servir la clàusula ALLOW FILTERING degut a l'impacte negatiu en el rendiment del clúster i perquè serà retirada en les properes versions de Cassandra.

```
cqlsh:crimes> SELECT sum(total) FROM
year_month_day_time_area_sub_area_crime_age_sex WHERE
crime_description = 'PICKPOCKET';

 system.sum(total)
-------------------
              2590

(1 rows)

Warnings :
Aggregation query used without partition key
```

Consulta 4 (20%)

Total de crims agrupats per sexe i filtrat pel nom d'àrea 'Hollywood' i sub_area amb codi '0643'

```
cqlsh:crimes> SELECT vict_sex, count
          ... FROM area_sub_area_desc_age_sex
          ... WHERE area_name = 'Hollywood' AND sub_area =
'0643';

 vict_sex | count
----------+-------
        F |     2
        M |     2
        X |     1
        F |     1

---MORE---
```

A continuació s'hauria de fer la agrupació i suma en la part del client.

Warning en consultes (5%)

Algunes de les consultes executades anteriorment retornen l'advertència *warning*: *Aggregation query used without partition key.* Què significa?

Això passa quan s'intenta filtrar per un atribut que no és part de la clau de partició, obligant Cassandra a buscar a través de totes les particions disponibles, ja que no té prou informació per saber en quines particions es troben específicament els registres buscats. Aquesta consulta és ineficient perquè pot involucrar moltes dades a través de múltiples nodes.

Detall de les taules usades (10%)

Per cada taula o agregat utilitzat per resoldre l'exercici, indica:

- Nom de taula, agregat utilitzat en la taula i consultes en què s'ha fet servir.
- Columnes que conformen la clau primària, clau de partició escollida.
- **El perquè de la clau primària i la partició (punt imprescindible per puntuar a la pregunta).**

Nota: contestar només amb un llistat de taules i columnes de clau primària i particions no puntua. El que puntua és la justificació de l'elecció de les columnes de clau primària i de partició.

Taula: area_sub_area_desc_age_sex

Consultes: Consulta 4

Columnes de la clau primària: ((area_name, sub_area), crime_description, vict_age, vict_sex)

Clau de partició: (area_name, sub_area)

Comentari: Les columnes area_name i sub_area les he utilitzat per definir la clau de partició ja que vull agrupar els registres per àea i sub-àrea, ja que així les consultes que tinguin a veure amb aquesta informació siguin més eficients. En canvi, crime_description, vict_age i vict_sex les utilitzo com a clustering per ordenar les entrades dintre de cada partició.

Taula: year_month_day_time_area_sub_area_crime_age_sex

Consultes: Consulta 1, Consulta 2, Consulta 3

Columnes de la clau primària: ((year_occ, month_occ), day_occ, time_occ, area_name, sub_area, crime_description, vict_age, vict_sex)

Clau de partició: (year_occ, month_occ)

Comentari: Les claus de partició són year_oss i month_occ, ja que així he pogut agrupar els registres per anys i mes, facilitant el fer consultes temporals (Consulta 1). day_occ, time_occ, area_name, sub_area, crime_description, vict_age, i vict_sex les utilitzo com a clusteriong per ordenar dintre de cada participació.

Nota: No ha sigut necessari carregar "year_month_day_time_area_sub_area_weapon.csv".

Exercici 2: Neo4j (35%)

Consulta 1 (10%)

Comptar i presentar el nombre de personatges que apareixen al llibre 7 d'aquesta història de ficció.

```
MATCH (p:Person)
WHERE 7 IN p.books
RETURN COUNT(p) AS numberOfCharacters
```

"numberOfCharacters"
71

Busquem totes les persones que tinguin el valor 7 a l'atribut books i en retornem el count.

Consulta 2 (10%)

Presentar el nom de la casa que ostenta un major nombre de cases lleials en la seva història i el nombre total de cases que li són o han estat lleials.

```
MATCH (h:House)<-[:SWORN_TO]-(loyalHouse:House)
RETURN h.name AS houseName, COUNT(loyalHouse) AS numberOfLoyalHouses
ORDER BY numberOfLoyalHouses DESC
LIMIT 1
```

"houseName"	"numberOfLoyalHouses"
"House Tyrell of Highgarden"	59

Primer busquem totes les cases que han jurat lleialtat a una altra casa, agrupem pel nom de la casa i comptem el nombre de cases lleials. Ordenem els resultats en ordre descendent i agafem la prima fila, que serà la casa amb més nombre de cases lleials.

Consulta 3 (15%)

Presentar el nom i les característiques de l'escut d'armes d'aquelles cases de menor rang (aquelles a les quals cap altra casa els ha jurat lleialtat), que incorporin en la descripció d'aquest escut un lleó (lion).

```
MATCH (h:House)
WHERE NOT (h)<-[:SWORN_TO]-(:House) AND h.coatOfArms CONTAINS ' lion '
RETURN h.name AS houseName, h.coatOfArms AS coatOfArmsDescription
```

"houseName"	"coatOfArmsDescription"
"House Lannister of Darry"	"Quarterly, a gold lion on a red field; a black plowman on a brown field(Quarterly, first and fourth gules a lion rampant or (for Lannister), second and third tenné a plowman sable (for Darry))"
"House Reyne of Castamere"	"A red lion rampant regardant with a forked tail, with gold teeth and claws, on a silver field(Argent, a lion rampant regardant queue-fourché gules, armed and langued or)"
"House Grandison of Grandview"	"A black sleeping lion, on a yellow field(Or, a lion dormant sable)"
"House Vikary"	"Quarterly: a red boar's head on a white field; beneath a gold bend sinister, a silver lion rampant regardant with a forked tail, with gold teeth and claws, on a red field."
"House Manning"	"A red sea lion between two black pallets on white"

Primer busquem totes les cases a les que cap altra casa els hi ha jurat lleialtat, mirem si en la descripció del seu escut d'armes hi apareix la paraula " lion " (amb espais, ja que si no també buscaria paraules que tenen "lion" en ella) i retorna el nom de la casa i la descripció.

Consulta 4 (15%)

Necessitem obtenir dades fundacionals sobre les cases. Per aconseguir-ho volem presentar agrupats, per any de creació, els noms de les cases fundades en aquest any. Es demana presentar els 2 anys amb major nombre de cases fundades no extintes (sense any d'extinció), el nombre de cases fundades en aquest any que encara perduren i una llista on s'agrupin els noms d'aquestes cases.

```
MATCH (h:House)
WHERE h.founded IS NOT NULL AND h.diedOut IS NULL
WITH h.founded AS year, collect(h.name) AS houses
WITH year, size(houses) AS count, houses
ORDER BY count DESC
LIMIT 2
RETURN year, count, houses;
```

"year"	"count"	"houses"
"Age of Heroes"	9	["House Stark of Winterfell","House Lannister of Casterly Rock","House Bracken of Stone Hedge","House Greyjoy of Pyke","House Blackwood of Raventree Hall","House Royce of Runestone","House Bolton of the Dreadfort","House Massey of Stonedance","House Casterly of Casterly Rock"]
"299 AC"	5	["House Lannister of Darry","House Frey of Riverrun","House Tyrell of Brightwater Keep","House Baelish of Harrenhal","House Foote of Nightsong"]

Busquem totes les cases que tenen un any de fundació diferent de NULL i que encara no han sigut extintes. Després les agrupem per any de fundació en una llista. Comptem quants elements hi ha a cada llista, els ordenem de forma descendent i agafem els dos primers.

Consulta 5 (15%)

Presentar les cases del regne que han sofert més de 3 escissions. És indiferent que les cases escindides s'hagin extingit o no. La consulta ha de retornar com a resultat el nom de la casa principal, el nombre de cases escindides i, en un sol camp agrupat, els noms de les cases escindides, ordenant els resultats descendentment per nombre de cases escindides.

```
MATCH (h:House)<-[:BRANCH_OF]-(b:House)
WITH h.name AS mainHouseName, collect(b.name) AS cadetBranches
WHERE size(cadetBranches) > 3
RETURN mainHouseName, size(cadetBranches) AS numberOfBranches,
cadetBranches
ORDER BY numberOfBranches DESC;
```

"mainHouseName"	"numberOfBranches"	"cadetBranches"
"House Lannister of Casterly Rock"	5	["House Lannett","House Lannister of Lannisport","House Lannister of Darry","House Lanny","House Lantell"]
"House Harlaw of Harlaw"	4	["House Harlaw of the Tower of Glimmering","House Harlaw of Grey Garden","House Harlaw of Harlaw Hall","House Harlaw of Harridan Hill"]
"House Baratheon of Storm's End"	4	["House Baratheon of Dragonstone","House Bolling","House Wensington","House Baratheon of King's Landing"]

Primer busquem totes les relacions “BRANCH_OF”, que ens indiquen que una casa “b” és una branca escindida d’una cada “h”. Un cop tenim aquestes relacions, agrupem les cases escindides amb la casa principal i creem una llista de totes les cases escindides per cada casa principal. Filtrem els grups per a que només ens quedin aquells on hi ha 3 o més cases escindides i retornem la llista de grups ordenats de manera descendent.

Consulta 6 (15%)

Presentar als 6 personatges amb major nombre de descendents que van arribar a ser governants de la seva casa, incloent-hi la dada de quants fills totals van tenir. La consulta ha d'incloure el nom del personatge, presentar la seva relació de parentiu (“mother”/”father”) amb els seus descendents, els noms d'aquests descendents (agrupats en un *array* de *strings*), el nombre de descendents inclosos en l’*array* (camp pel qual s'ordenarà el resultat) i el nombre total de fills del personatge (no tots tenen per què haver estat governants, ni tan sols tenir dret a governar si són “il·legítims”). El resultat es presentarà en ordre descendent pel nombre de descendents governants i, en cas d'igualtat, en ordre descendent per la quantitat de fills totals.

```
MATCH (p:Person)-[:PARENT_OF]->(desc:Person)
OPTIONAL MATCH (h:House)-[:LED_BY]->(desc)
WITH p,
     COLLECT(DISTINCT desc.name) AS children_names,
     COLLECT(DISTINCT CASE WHEN h IS NOT NULL THEN desc.name
ELSE NULL END) AS governing_children,
     COUNT(DISTINCT CASE WHEN h IS NOT NULL THEN desc ELSE NULL
END) AS descendant_count,
     SIZE((p)-[:PARENT_OF]->(:Person)) AS total_children
RETURN p.name AS name,
       governing_children AS descendants,
       descendant_count,
       total_children
ORDER BY descendant_count DESC, total_children DESC
LIMIT 6
```

"name"	"descendants"	"descendant_count"	"total_children"
"Steffon Baratheon"	["Stannis Baratheon"]	1	2
"Robert I Baratheon"	["Tommen Baratheon"]	1	2
"Quellon Greyjoy"	["Euron Greyjoy"]	1	2
"Cersei Lannister"	["Tommen Baratheon"]	1	2
"Cassana Estermont"	["Stannis Baratheon"]	1	2
"Aegon III"	[]	0	3

Primer trobem totes les relacions on una persona és pare/mare d’una altra persona. Després busquem si alguns dels descendents lidera una casa. Aquesta segona part és opcional perquè es pot donar el cas que no tots els descendents liderin una casa.

A continuació agrupem els resultats i posem els noms de tots els descendents en una llista “children_names”. Si els descendents governen una casa els afegim a la llista “governing_children”. Comptem quants descendents governen una casa i guardem el valor. També comptem el nombre total de descendents.

Ja podem fer el retorn amb el nom, la llista de noms dels descendents que governen una casa, el nombre de descendents que governen una casa i el nombre total de descendents. Ordenen en descendent i agafem els primers 6.

Consulta 7 (20%)

Dels resultats de la consulta anterior ens ha cridat l'atenció el cas de "Robert I Baratheon". Es desitja obtenir informació sobre els seus fills, per la qual cosa necessitem una consulta que ens presenti, per cadascun dels seus fills: el nom del fill/filla, la seva data de naixement i defunció (si existeixen), els seus títols i els seus àlies. Per als fills/filles que no tinguin títols, àlies o no hagin mort, s'haurà de mostrar en la columna respectiva els següents continguts: “sense títols” (per als que no tinguin títols), “sense àlies” (per als que no tinguin àlies i “encara viu” (per als que encara no han mort).

```
MATCH (robert:Person {name: "Robert I Baratheon"})-[:PARENT_OF]-
>(child:Person)
RETURN child.name AS name,
       COALESCE(child.born, "data de naixement desconeguda") AS
born,
       COALESCE(child.died, "encara viu") AS died,
       CASE
           WHEN size(child.titles) = 0 THEN "sense títols"
           ELSE apoc.coll.toSet(child.titles)
       END AS titles,
       CASE
           WHEN size(child.aliases) = 0 THEN "sense àlies"
           ELSE apoc.coll.toSet(child.aliases)
       END AS aliases
```

"name"	"born"	"died"	"titles"	"aliases"
"Tommen Baratheon"	"291 AC"	"encara viu"	["King of the Andals, the Rhoynar and the First Men","Lord of the Seven Kingdoms"]	["The Boy King"]
"Joffrey Baratheon"	"286 AC, at King's Landing"	"300 AC, at Red Keep, King's Landing"	["King of the Andals, the Rhoynar and the First Men","Lord of the Seven Kingdoms","Protector of the Realm"]	["Joffrey the Illborn","The Young Usurper","Aerys the Third","Joffrey-called-Baratheon"]

Primer busquem el node “Person” amb el nom “Robert I Baratheon” i els seus fills. A continuació es fan servir COALESCE i CASE per gestionar els valors NULL i arrays buits respectivament. Si fos el cas que la data de naixement és NULL, s’imprimirà el missatge “data de naixement desconeguda”, el mateix per la data de la mort, però imprimint “encara viu”. Si la mida de la llista de títols és 0, aleshores s’imprimeix “sense títols”, en cas contrari, imprimim la llista de títols. Repetim el mateix pels àlies.

Exercici 3: MongoDB (35%)

Consulta (15%)

Com que no hi ha una col·lecció amb tots els possibles valors de les qualificacions de les pel·lícules (*Rating*), ni una restricció d'integritat relacional amb els valors possibles, volem obtenir els valors únics de les qualificacions que hi ha en la col·lecció *Sakila_films*. La consulta ha de retornar solament els valors únics (no tota la llista completa de totes les classificacions de totes les pel·lícules) i per ordre alfabètic descendent. No ha de retornar el *_id* del document.

```
> db.Sakila_films.distinct("Rating").sort().reverse()
[ "R", "PG-13", "PG", "NC-17", "G" ]
```

Amb “distinc” obtenim tots els valors únics del camp “Rating”, els ordenem amb sort() i invertim l’ordre dels elements amb reverse().

Consulta (10%)

Ara que sabem els codis de qualificació, volem obtenir el recompte de les pel·lícules que no són aptes per a menors de 17 anys (valor de qualificació "NC-17"). La consulta ha de retornar només la xifra.

```
> db.Sakila_films.countDocuments({Rating: "NC-17"})
210
```

Amb aquesta comanda fem el recompte de tots els documents de la col·lecció que tenen el valor de “Rating” exactament en NC-17.

Consulta (15%)

Volem saber els 3 actors que han participat en més pel·lícules. S'ha de mostrar el recompte dels actors amb el seu identificador i després un llistat d'aquests tres actors mostrant el nom i el cognom. És possible que aquest exercici no es pugui resoldre amb una sola consulta, en la primera haureu d'obtenir el recompte i en la segona obtenir el nom i cognom.

Suggerència: Per agrupar correctament els actors en la col·lecció de pel·lícules s’ha de fer servir el camp actorId.

Nota: El llistat dels tres actors no és necessari que estigui ordenat.

```
> db.Sakila_films.aggregate([
...     { $unwind: "$Actors" },
...     { $group: { _id: "$Actors.actorId", count: { $sum: 1 } }
},
...     { $sort: { count: -1 } },
...     { $limit: 3 }
... ])
{ "_id" : 107, "count" : 42 }
{ "_id" : 102, "count" : 41 }
{ "_id" : 198, "count" : 40 }

> db.Sakila_actors.find(
...     { _id: { $in: [107, 102, 198] } },
...     { _id: 0, FirstName: 1, LastName: 1 }
... )
{ "FirstName" : "WALTER", "LastName" : "TORN" }
{ "FirstName" : "GINA", "LastName" : "DEGENERES" }
{ "FirstName" : "MARY", "LastName" : "KEITEL" }
```

Per obtenir la informació hem fet dos passos:

- Pas 1: Obtenció del recompte de pel·lícules per actor i retorn dels “id”.
- Pas 2: Obtenció dels noms i cognoms dels actors de les “id” del pas 1.

Consulta (15%)

Volem saber les pel·lícules en les quals han treballat més actors. La consulta ha de retornar un únic registre o document amb el nom de la pel·lícula i el recompte d'actors.

```
> db.Sakila_films.aggregate([
...     { $unwind: "$Actors" },
...     { $group: { _id: "$Title", actorCount: { $sum: 1 } } },
...     { $sort: { actorCount: -1 } },
...     { $limit: 1 }
... ])
{ "_id" : "LAMBS CINCINATTI", "actorCount" : 15 }
```

Primer fem servir unwind per descomposar l’array de vectors en diferents documents. Agrupem pel títol de la pel·lícula i comptem el nombre d’actors en cada una. Finalment ordenem els resultats de forma descendent i agafem el primer resultat.

Consulta (20%)

Per a tots els lloguers dels clients volem afegir una valoració numèrica de les pel·lícules. Els valors aniran del 0 al 5 i s'utilitzarà el valor -1 per a aquells clients que no han fet cap valoració. El nou camp a afegir es dirà "Rate" i com és lògic s'inicialitzarà a -1.

S'ha d'adjuntar:

- La comanda d'actualització juntament amb el seu resultat.

```
> db.Sakila_customers.updateMany(
...     {},
...     { $set: { "Rentals.$[].Rate": -1 } }
... )
{ "acknowledged" : true, "matchedCount" : 599, "modifiedCount" : 599 }
```

- Una consulta que mostri la llista dels lloguers amb el camp afegit **del desè client de la col·lecció ordenada per cognom ascendentment**. La consulta de comprovació només ha de retornar el cognom (camp "Last Name") i la llista de lloguers del client en qüestió.

```
> db.Sakila_customers.aggregate([
...     { $sort: { "Last Name": 1 } },
...     { $skip: 9 },
...     { $limit: 1 },
...     {
...         $project: {
...             _id: 0,
...             "Last Name": 1,
...             Rentals: {
...                 $map: {
...                     input: "$Rentals",
...                     as: "rental",
...                     in: {
...                         ID: "$$rental.filmId",
...                         rentalId: "$$rental.rentalId",
...                         staffId: "$$rental.staffId",
...                         "Film Title": "$$rental.Film Title",
...                         "Rental Date": "$$rental.Rental Date",
...                         "Return Date": "$$rental.Return Date",
...                         Rate: "$$rental.Rate"
...                     }
...                 }
...             }
...         }
...     }
... ])
```

{ "Last Name" : "ANDREWS", "Rentals" : [{ "ID" : 153, "rentalId" : 1279, "staffId" : 2, "Film Title" : "CITIZEN SHREK", "Rental Date" : "2005-06-15 08:13:57.0", "Return Date" : "2005-06-18 09:36:57.0", "Rate" : -1 }, { "ID" : 302, "rentalId" : 3381, "staffId" : 2, "Film Title" : "FANTASIA PARK", "Rental Date" : "2005-06-21 14:02:59.0", "Return Date" : "2005-06-29 18:11:59.0", "Rate" : -1 }, { "ID" : 557, "rentalId" : 3869, "staffId" : 1, "Film Title" : "MANCHURIAN CURTAIN", "Rental Date" : "2005-07-06 17:56:46.0", "Return Date" : "2005-07-10 20:44:46.0", "Rate" : -1 }, { "ID" : 641, "rentalId" : 4134, "staffId" : 1, "Film Title" : "ORANGE GRAPES", "Rental Date" : "2005-07-07 08:14:24.0", "Return Date" : "2005-07-09 10:42:24.0", "Rate" : -1 }, { "ID" : 69, "rentalId" : 4157, "staffId" : 1, "Film Title" : "BEVERLY OUTLAW", "Rental Date" : "2005-07-07 09:04:26.0", "Return Date" : "2005-07-08 09:55:26.0", "Rate" : -1 }, { "ID" : 966, "rentalId" : 5069, "staffId" : 2, "Film Title" : "WEDDING APOLLO", "Rental Date" : "2005-07-09 04:56:30.0", "Return Date" : "2005-07-13 23:53:30.0", "Rate" : -1 }, { "ID" : 26, "rentalId" : 5756, "staffId" : 2, "Film Title" : "ANNIE IDENTITY", "Rental Date" : "2005-07-10 12:39:28.0", "Return Date" : "2005-07-11 14:08:28.0", "Rate" : -1 }, { "ID" : 166, "rentalId" : 6569, "staffId" : 2, "Film Title" : "COLOR PHILADELPHIA", "Rental Date" : "2005-07-12 05:47:40.0", "Return Date" : "2005-07-20 06:23:40.0", "Rate" : -1 }, { "ID" : 39, "rentalId" : 7359, "staffId" : 2, "Film Title" : "ARMAGEDDON LOST", "Rental Date" : "2005-07-27 14:51:04.0", "Return Date" : "2005-07-31 16:03:04.0", "Rate" : -1 }, { "ID" : 277, "rentalId" : 9818, "staffId" : 1, "Film Title" : "ELEPHANT TROJAN", "Rental Date" : "2005-07-31 11:34:32.0", "Return Date" : "2005-08-04 08:20:32.0", "Rate" : -1 }, { "ID" : 115, "rentalId" : 11386, "staffId" : 1, "Film Title" : "CAMPUS REMEMBER", "Rental Date" : "2005-08-02 18:24:03.0", "Return Date" : "2005-08-06 21:22:03.0", "Rate" : -1 }, { "ID" : 610, "rentalId" : 12451, "staffId" : 1, "Film Title" : "MUSIC BOONDOCK", "Rental Date" : "2005-08-18 11:04:42.0", "Return Date" : "2005-08-20 08:20:42.0", "Rate" : -1 }, { "ID" : 6, "rentalId" : 12764, "staffId" : 1, "Film Title" : "AGENT TRUMAN", "Rental Date" : "2005-08-18 23:14:15.0", "Return Date" : "2005-08-22 20:23:15.0", "Rate" : -1 }, { "ID" : 356, "rentalId" : 13482, "staffId" : 1, "Film Title" : "GIANT TROOPERS", "Rental Date" : "2005-08-20 01:14:30.0", "Return Date" : "2005-08-24 04:57:30.0", "Rate" : -1 }, { "ID" : 949, "rentalId" : 382, "staffId" : 1, "Film Title" : "VOLCANO TEXAS", "Rental Date" : "2005-05-27 10:12:00.0", "Return Date" : "2005-05-31 15:03:00.0", "Rate" : -1 }, { "ID" : 155, "rentalId" : 2188, "staffId" : 2, "Film Title" : "CLEOPATRA DEVIL", "Rental Date" : "2005-06-18 01:19:04.0", "Return Date" : "2005-06-25 03:59:04.0", "Rate" : -1 }, { "ID" : 745, "rentalId" : 2471, "staffId" : 2, "Film Title" : "ROSES TREASURE", "Rental Date" : "2005-06-18 20:31:00.0", "Return Date" : "2005-06-24 18:01:00.0", "Rate" : -1 }, { "ID" : 827, "rentalId" : 6472, "staffId" : 1, "Film Title" : "SPICE SORORITY", "Rental Date" : "2005-07-12 01:33:25.0", "Return Date" : "2005-07-15 20:26:25.0", "Rate" : -1 }, { "ID" : 960, "rentalId" : 9672, "staffId" : 2, "Film Title" : "WARS PLUTO", "Rental Date" : "2005-07-31 06:34:06.0", "Return Date" : "2005-08-08 10:29:06.0", "Rate" : -1 }, { "ID" : 757, "rentalId" : 9931, "staffId" : 2, "Film Title" : "SAGEBRUSH CLUELESS", "Rental Date" : "2005-07-31 15:18:19.0", "Return Date" : "2005-08-04 14:23:19.0", "Rate" : -1 }, { "ID" : 673, "rentalId" : 10620, "staffId" : 1, "Film Title" : "PERSONAL LADYBUGS", "Rental Date" : "2005-08-01 15:09:17.0", "Return Date" : "2005-08-09 13:58:17.0", "Rate" : -1 }, { "ID" : 416, "rentalId" : 12831, "staffId" : 1, "Film Title" : "HIGHBALL POTTER", "Rental Date" : "2005-08-19 01:40:43.0", "Return Date" : "2005-08-28 05:22:43.0", "Rate" : -1 }, { "ID" : 145, "rentalId" : 13536, "staffId" : 2, "Film Title" : "CHISUM BEHAVIOR", "Rental Date" : "2005-08-20 03:35:16.0", "Return Date" : "2005-08-25 04:06:16.0", "Rate" : -1 }] }

Agafem la col·lecció “Sakila_customers”, ordenem els clients pel seu cognom, en saltem 10 (ja que comença per 0) i retornem el cognom i la llista de les seves pel·lícules llogades. A la llista hi hem afegit tota la informació.

Consulta (25%)

Ara mateix tenim la valoració de cada lloguer correctament enregistrada. No obstant això, cada vegada que vulguem obtenir la valoració mitjana d'una pel·lícula haurem de fer una consulta a la col·lecció de *Sakila_customers* i calcular la valoració mitjana de tots els lloguers d'aquesta pel·lícula. Aquest procés és molt costós, tenint en compte que hi haurà moltes més consultes que modificacions, per la qual cosa se'ns sol·licita que dissenyem una solució més eficient.

L'aplicació que modifica les dades cada vegada que es produeix un lloguer, és a dir, l'aplicació que executa la consulta per actualitzar la col·lecció de *Sakila_customers*, està capacitada per executar més consultes o comandes en el moment de l'actualització. Per tant, podem mantenir actualitzada la puntuació mitjana d'una pel·lícula cada vegada que s'introdueix una valoració nova (excloent en el càlcul les puntuacions amb valor -1 que significa “no puntuació”). En cas d'inconsistència temporal (per exemple a causa d'una modificació realitzada per una altra aplicació) es podrien tornar a recalcular tots els valors en un procés de recàlcul de totes les mitjanes de totes les pel·lícules (aquest cas no s'aborda en aquest exercici).

Se'ns sol·licita que realitzem aquesta implementació i dissenyem les consultes a executar per mantenir la mitjana actualitzada en la col·lecció que hagueu triat. Amb aquest objectiu en ment, resol les següents dues preguntes:

Pregunta 1: En quina col·lecció desaries l’atribut per mantenir el valor calculat?

La desaria a la col·lecció “Sakira_films”, ja que la mitjana de les valoracions està directament relacionada amb la pel·lícula específica. Emmagatzemar aquesta informació d’aquesta manera permet un accés ràpid i eficient a la mitjana de la valoració sense necessitat de consultar altres col·leccions.

Pregunta 2: Es rep un lloguer de la pel·lícula amb títol "AFFAIR PREJUDICE" i "filmId" 4 amb un nou lloguer en l'array Rentals del client amb identificador 1. En aquest cas, el programa haurà de calcular la nova mitjana d'aquesta pel·lícula i guardar el valor on hagueu decidit en el punt anterior. Per realitzar l'exercici, haureu d'inserir el lloguer amb la valoració usant la següent instrucció:

```
db.Sakila_customers.updateOne(
      { "_id": 1 },
      { $push:
      {
            "Rentals":
            {
                  "filmId": 4,
                  "rentalId": 10000,
                  "staffId": 2,
                  "Film Title": "AFFAIR PREJUDICE",
                  "Payments": [
                  {
                        "Amount": 10.7,
                        "Payment Date": "2024-01-01 00:00:12.0",
                        "Payment Id": 3
                  }
                  ],
                  "Rental Date": "2023-12-29 00:00:12.0",
                  "Return Date": "2024-01-01 00:00:12.0",
                  "Rate": 3
            }
      }
      }
);
```

Es demanen dues consultes:

- La consulta per obtenir la qualificació mitjana d'aquesta pel·lícula tenint en compte el registre inserit en la instrucció anterior.

- La consulta per actualitzar el valor en la col·lecció que heu triat en la pregunta anterior. Com que MongoDB és flexible, no cal que estiguin tots els documents de la col·lecció amb aquest atribut i podem assumir que aquest atribut s'anirà afegint a tots els documents a mesura que es vagin produint actualitzacions en les qualificacions de les pel·lícules.

Per facilitar la resolució hem dividit tota la feina en diferents passos, que aprofitarem per explicar la lògica.

Pas 1: Inserim una nova puntuació amb el codi facilitat a l'enunciat de l'activitat.

Pas 2: Calculem la nova mitjana de valoració de la pel·lícula:

```
> var averageRatingPipeline = [
...     { $unwind: "$Rentals" },
...     { $match: { "Rentals.filmId": 4, "Rentals.Rate": { $ne: -1 } } },
...         { $group: { _id: "$Rentals.filmId", averageRating: { $avg:
"$Rentals.Rate" } } }
... ];
>
>                 var                 averageRatingResult                 =
db.Sakila_customers.aggregate(averageRatingPipeline).toArray();
>
>    var    averageRating    =    averageRatingResult.length    >    0    ?
averageRatingResult[0].averageRating : -1;
```

Explicació:

Definim el pipeline d'agrupació, que consta de tres parts:

- Descomposició de l'array Rentals en múltiples documents, així es pot tractar cada lloguer per separat.
- Filtrem els filmId per només tindre en compte els que tenen valor 4 i una valoració diferent de -1.
- Agrupem els documents i calculem la mitjana de valoracions.

  A continuació executem el pipeline i mirem si l'array "averageRatingResult" té algun document, si és el cas, "averageRating" és el valor de la mitjana, en cas contrari es defineix com a -1.

Pas 3: Actualització de la col·lecció amb el nou valor:

```
> db.Sakila_films.updateOne(
...     { _id: 4 },
...     { $set: { averageRating: averageRating } }
... );
{ "acknowledged" : true, "matchedCount" : 1, "modifiedCount" : 1
}
```

Fem servir updateOne per només actualitzar un sol document, en el nostre cas el que te "_id" igual a 4. Finalment s'actualitza el valor de la variable.

Pas 4: Comprovació de la nova mitjana de valoracions

```
> db.Sakila_films.findOne(
...     { _id: 4 },
...     { _id: 0, Title: 1, averageRating: 1 }
... );
{ "Title" : "AFFAIR PREJUDICE", "averageRating" : 3 }
```

Busquem la pel·lícula que hem actualitzat i mirem que el seu averageRating s'ha vist modificat.