Práctica 2: Cassandra

Martínez Ostoa Néstor Iván
Bases de Datos No Estructuras - 0600
Ciencia de Datos, IIMAS, UNAM

Mayo 2021

1. Introducción

Para esta práctica empleé la base de datos no estructurada Cassandra para administrar una plataforma de libros [1]. Cassandra es una base de datos no estructuradas con un enfoque en la descentralización de la información aparentando una centralización al usuario. Esto significa que la información que almacena Cassandra se encuentra distribuida entre diferentes nodos que podrían estar en diversas partes del mundo. Para realizar esto y poder seguir siendo altamente escalable y veloz, Cassandra ofrece las siguientes características:

- **Distribuida y Descentralizada**: todos la información opera sobre múltiples computadoras y servidores aparentando ser una sola computadora. Aunado a esto, todos los todos parecen los mismos mediante el protocolo *gossip* el cual ayuda a que Cassandra no tenga posible punto de fallo físico
- **Escalabilidad elástica**: Cassandra es capaz de agregar y quitar más servidores/computadoras según sea necesario
- **Alta disponibilidad y tolerante a fallos**: Cassandra permite agregar más nodos para sustituir a los fallidos al instante (*no downtime*)

[1] La descripción de esta plataforma se da en la sección 2

- **Consistencia ajustable**: Cassandra puede ajustar el nivel de consistencia de la información en función del nivel de disponibilidad deseado. De acuerdo al teorema CAP de Brewer (*Consistency*, *Availability*, *Partition Tolerance*), Cassandra se encuentra del lado *AP* por lo que da prioridad a la disponibilidad y a la tolerancia a particiones, pero la consistencia puede ser ajustable a expensas de estos otros dos elementos
- **CQL**: Cassandra provee de su propio lenguaje de consulta (*Cassandra Query Language*) el cual comparte muchas similitudes con *SQL* pero está adaptado a funcionar con el modelo de datos de Cassandra. Ejemplo de operaciones básicas en Cassandra utilizando *CQL*:

```
cqlsh > CREATE KEYSPACE books_keyspace WITH replication = {'class': '
    SimpleStragegy', 'replication_factor': 1};

cqlsh > USE books_keyspace;

cqlsh > CREATE TABLE book (
    title text,
    authors set<text>,
    pages smallint,
    category text,
    PRIMARY KEY (title)
) WITH CLUSTERING ORDER BY (category ASC);
```

1.1. Modelo de datos

El modelo de datos de Cassandra está orientado a las columnas, por eso es que a Cassandra se le conoce como una base de datos columnar. Cada registro dentro de Cassandra se identifica por medio de una llave primaria compuesta. Dicha llave primaria contiene lo siguiente:

- Llave de partición (*partition key*): determine los nodos dentro de la base de datos en donde los renglones serán almacenados. Para determinar el nodo concreto, se aplica una función hash al campo indicado como *partition key*
- Llave de agrupamiento (*clustering key*): controla principalmente el orden de la información dentro de una partición

En la siguiente imagen muestro un diagrama del modelo de datos de Cassandra en un nivel más general y donde se especifica físicamente la funcionalidad de las particiones junto con los otros elementos del modelo de datos.

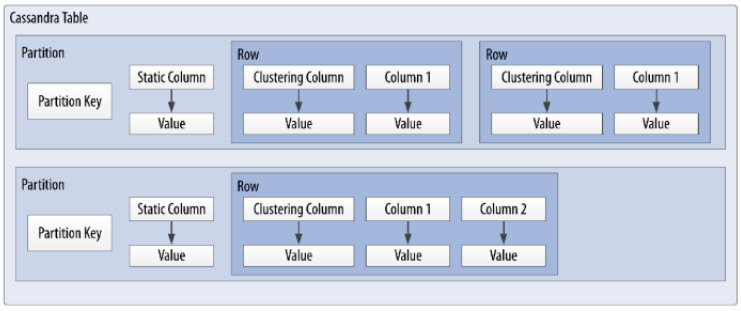

Figura 1: Elementos del modelo de datos de Cassandra

El modelo de datos de Cassandra tiene la siguiente jerarquía:

1. ***Cluster***: también llamado anillo (*ring*) es un elemento que contiene nodos a los cuales Cassandra, por medio de una función hash, asigna información
2. ***Keyspace***: es la analogía directa con una base de datos dentro de una manejador relacional de bases de datos por lo que sirve como un contenedor para tablas
3. ***Table***: las tablas en Cassandra contienen particiones y están agrupadas dentro de *keyspaces*. Se puede pensar como un contenedor para renglones organizadas por particiones
4. ***Partition***: las particiones se agrupan dentro de una misma tabla y cada partición contiene un renglón. Las particiones son identificadas unívocamente por medio de las llaves de partición
5. ***Row***: los renglones están organizados en particiones y son asignados a nodos dentro de un *cluster* por medio de la llave de partición y llave de agrupamiento
6. ***Column***: elemento más primitivo dentro del modelo de datos. Consta de una llave (nombre de la columna) y un valor

1.2. Modelado de datos - Diferencias con RDBMS

Finalmente, un aspecto fundamental de Cassandra es el estilo de modelado de datos. Para desarrolladores provenientes de bases de datos relacionales, es muy natural pensar en llaves foráneas, *joins* y normalización de tablas. Sin embargo, el modelado de datos en Cassandra (principalmente por ser una base de datos distribuida y descentralizada) se lleva de manera diferente.

Una característica esencial del modelado de datos en Cassandra es que se realiza pensando primero en las consultas a responder. Es decir, primero se piensa en las preguntas de negocio a responder (por ejemplo: .ºbtén los mejores libros por categoría") y posteriormente se diseñan las tablas teniendo en mente la consulta a la que se quiere responder. Formalmente, se podría decir que la mayoría de las tablas de Cassandra deberían responder a una pregunta de negocio en particular. Este estilo de modelado no es casualidad pues de acuerdo con [2], Cassandra fue diseñada originalmente para resolver un problema muy particular de Facebook: búsqueda de mensajes en la aplicación de Messenger.

Con base en lo anterior, las siguientes características se deben de tener en mente a la hora de modelar un problema para Cassandra:

- Cassandra no está optimizado para hacer *joins* entre tablas
- Cassandra no existe formalmente la integridad referencial
- En lugar de pensar en normalización, en Cassandra se debe de pensar en la denormalización de la información. Es decir, duplicar información es más eficiente siempre y cuando de una misma tabla se pueda obtener toda la información deseada en lugar de tener que consultar múltiples tablas y particiones.
- Empezar siempre primero con el modelado de las preguntas de negocio (*query-first design*)

- Diseñar para almacenamiento óptimo. Las tablas de Cassandra se almacenan en diversos archivos de disco por lo que se debe minimizar el número de particiones a buscar para satisfacer una consulta

2. Desarrollo

2.1. Objetivo y Alcance

El objetivo de esta práctica es desarrollar una plataforma de libros con las siguientes características:

- La plataforma debe mostrar las relaciones entre libros, clientes y calificaciones para cada libro
- Cada cliente debe tener información asociada (nombre, país, membresía, etc.)
- Cada cliente debe asignar una categoría para cada uno de sus libros
- Todos deben participar como clientes

Aunado a esto, los administradores de la base de datos deberán poder buscar lo siguiente:

- La categoría preferida de un cliente dado
- Obtener los clientes que más disfrutaron un libro dado
- Los mejores libros de una categoría dada

2.2. Esbozo de solución

Para implementar los requisitos descritos en la sección anterior (sección 2.1) seguí un enfoque orientado a consultas por lo que diseñe una tabla específica para cada una de las consultas más relevantes (descritas en la sección 2.2.1) del negocio.

2.2.1. Consultas

Las consultas pilar dentro de mi aplicación son las siguientes:

- **Q1**: Obtener los libros por título
- **Q2**: Obtener los libros por nombre de usuario
- **Q3**: Obtener los mejores libros de una categoría dada [2]
- **Q4**: Obtener los libros por calificación
- **Q5**: Obtener los clientes por nombre de usuario
- **Q6**: Obtener los clientes por país

[2] Las consultas marcadas en color rojo son las consultas pedidas por el negocio hacía los administradores (sección 2.1)

- **Q7**: Obtener los clientes por membresía
- Q8: Obtener la categoría preferida de un cliente dado
- Q9: Obtener los clientes que más disfrutaron un libro

2.2.2. Diagrama de Chebtoko - Consultas

A continuación se muestra un esquema de Chebtoko de las consultas y la relación entre ellas:

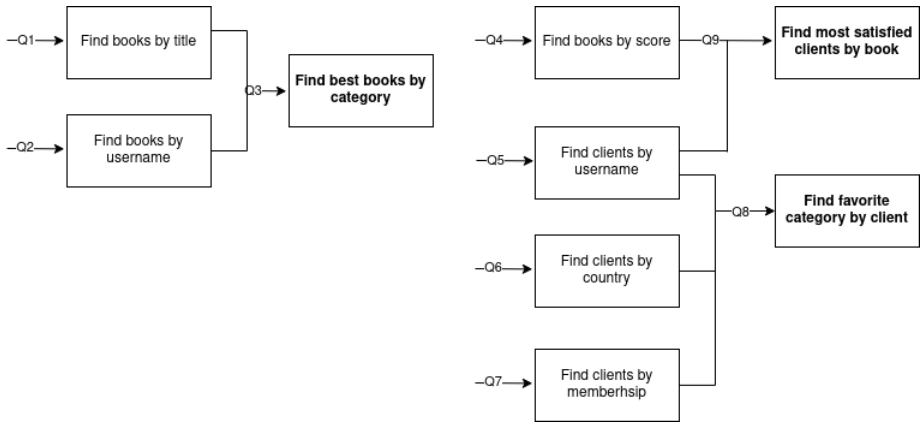

Figura 2: Diagrama de Chebtoko para las consultas pilar

2.2.3. Diagrama de Chebtoko - Keyspaces

Derivado del diagrama mostrado en la figura 2, realicé el siguiente diagrama de Chebtoko a nivel de *keyspaces* y el diagrama resultante es el que ocupé para la implementación física del modelo.

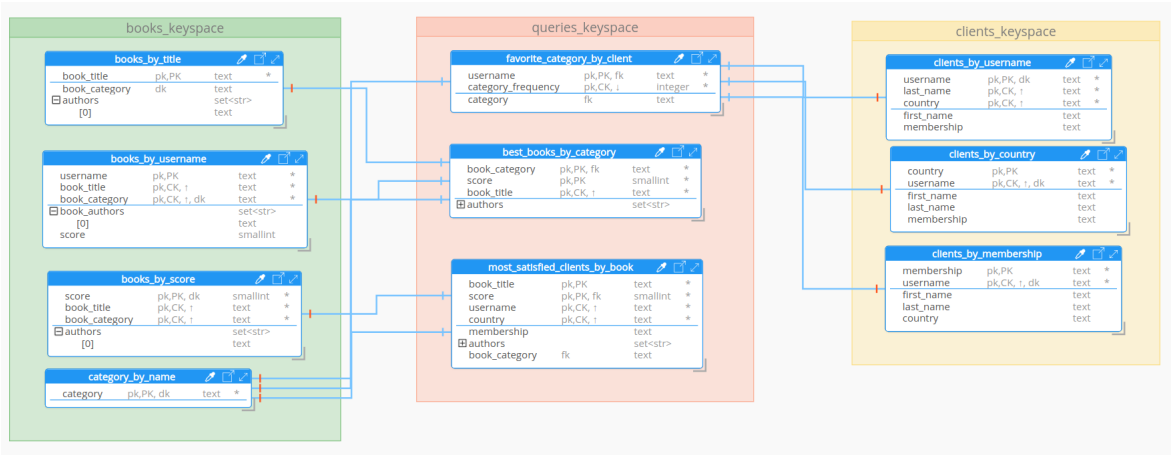

Figura 3: Diagrama de Chebtoko a nivel de *keyspaces* para las consultas pilar

2.3. Implementación - Definición de datos

Para la implementación física implementé un script [3] dependiendo del *keyspace* en cuestión:

- ***books_keyspace***:

```
CREATE KEYSPACE IF NOT EXISTS "books_keyspace"
  WITH REPLICATION = {
    'class' : 'SimpleStrategy',
    'replication_factor' : 1
  }
AND DURABLE_WRITES = false;

USE "books_keyspace";

CREATE TABLE IF NOT EXISTS "books_keyspace"."books_by_title" (
  "book_title" text,
  "book_category" text,
  "authors" set<text>,
  PRIMARY KEY ("book_title")
);

CREATE TABLE IF NOT EXISTS "books_keyspace"."books_by_username" (
  "username" text,
  "book_title" text,
  "book_category" text,
  "book_authors" set<text>,
  "score" smallint,
  PRIMARY KEY ("username", "book_title", "book_category")
)
WITH CLUSTERING ORDER BY ("book_title" ASC, "book_category" ASC);

CREATE TABLE IF NOT EXISTS "books_keyspace"."books_by_score" (
  "score" smallint,
  "book_title" text,
  "book_category" text,
  "authors" set<text>,
  PRIMARY KEY ("score", "book_title", "book_category")
)
WITH CLUSTERING ORDER BY ("book_title" ASC, "book_category" ASC);
```

- ***queries_keyspace***:

```
CREATE KEYSPACE IF NOT EXISTS "queries_keyspace"
  WITH REPLICATION = {
    'class' : 'SimpleStrategy',
    'replication_factor' : 1
  }
AND DURABLE_WRITES = false;

USE "queries_keyspace";

CREATE TABLE IF NOT EXISTS "queries_keyspace"."best_books_by_category" (
  "book_category" text,
  "book_title" text,
```

[3] Se pueden consultar aquí: https://github.com/nestorivanmo/iimas-data-science/tree/master/spring-21/bne/practicas/practica_cassandra/scripts

```
  "score" smallint,
  "authors" set<text>,
  PRIMARY KEY (("book_category", "score"), "book_title")
)
WITH CLUSTERING ORDER BY ("book_title" ASC);

CREATE TABLE IF NOT EXISTS "queries_keyspace"."most_satissfied_clients_by_book"
    (
  "book_title" text,
  "score" smallint,
  "username" text,
  "country" text,
  "membership" text,
  "authors" set<text>,
  PRIMARY KEY (("book_title", "score"), "username", "country")
)
WITH CLUSTERING ORDER BY ("username" ASC, "country" ASC);

CREATE TABLE IF NOT EXISTS "queries_keyspace"."favorite_category_by_client" (
  "username" text,
  "category_frequency" int,
  "category" text,
  PRIMARY KEY ("username", "category_frequency")
)
WITH CLUSTERING ORDER BY ("category_frequency" DESC);
```

- ***clients_keyspace***:

```
CREATE KEYSPACE IF NOT EXISTS "clients_keyspace"
  WITH REPLICATION = {
    'class' : 'SimpleStrategy',
    'replication_factor' : 1
  }
AND DURABLE_WRITES = false;

USE "clients_keyspace";

CREATE TABLE IF NOT EXISTS "clients_keyspace"."clients_by_username" (
  "username" text,
  "first_name" text,
  "last_name" text,
  "country" text,
  "membership" text,
  PRIMARY KEY ("username", "last_name", "country")
)
WITH CLUSTERING ORDER BY ("last_name" ASC, "country" ASC);

CREATE TABLE IF NOT EXISTS "clients_keyspace"."clients_by_country" (
  "country" text,
  "username" text,
  "first_name" text,
  "last_name" text,
  "membership" text,
  PRIMARY KEY ("country", "username")
)
WITH CLUSTERING ORDER BY ("username" ASC);

CREATE TABLE IF NOT EXISTS "clients_keyspace"."clients_by_membership" (
```

```
  "membership" text,
  "username" text,
  "first_name" text,
  "last_name" text,
  "country" text,
  PRIMARY KEY ("membership", "username")
)
WITH CLUSTERING ORDER BY ("username" ASC);
```

2.4. Implementación - Inserción de datos

Para la inserción de datos simulé el proceso que seguiría la aplicación soportada por esta base de datos. El script principal es un notebook de Python que simula el proceso por cada script. Aunado a esto, el script se encarga de generar las sentencias de inserción de datos según los *keyspaces* de la sección anterior (2.3). Dicho script se llama `books_cleaning.ipynb` y se puede encontrar en la siguiente URL: https://bit.ly/2RM9P9w. Los scripts generados por `books_cleaning.ipynb` se encuentran en esta URL: https://bit.ly/2RIPB0d.

A continuación se muestran las capturas de pantalla de la implementación física del modelo mostrado en la figura 3.

- Creación de *keyspaces*

```
cqlsh:queries_keyspace> DESCRIBE KEYSPACES;

books_keyspace  system_auth  queries_keyspace    system_traces
system_schema   system       system_distributed  clients_keyspace
```

Figura 4: Creación de *keyspaces*

```
cqlsh:queries_keyspace> DESCRIBE books_keyspace;

CREATE KEYSPACE books_keyspace WITH replication = {'class': 'SimpleStrategy', 'replication_factor': '1'}  AND durable_writes = false;
```

Figura 5: Creación de *books_keyspace*

```
cqlsh:queries_keyspace> DESCRIBE queries_keyspace;

CREATE KEYSPACE queries_keyspace WITH replication = {'class': 'SimpleStrategy', 'replication_factor': '1'}  AND durable_writes = false;
```

Figura 6: Creación de *queries_keyspace*

```
cqlsh:queries_keyspace> DESCRIBE clients_keyspace;

CREATE KEYSPACE clients_keyspace WITH replication = {'class': 'SimpleStrategy', 'replication_factor': '1'}  AND durable_writes = false;
```

Figura 7: Creación de *clients_keyspace*

- Creación de tablas por *keyspace* + inserción de información

- *books_keyspace*:

```
cqlsh:books_keyspace> DESCRIBE books_by_title;

CREATE TABLE books_keyspace.books_by_title (
    book_title text PRIMARY KEY,
    authors set<text>,
    book_category text
) WITH bloom_filter_fp_chance = 0.01
    AND caching = {'keys': 'ALL', 'rows_per_partition': 'NONE'}
    AND comment = ''
    AND compaction = {'class': 'org.apache.cassandra.db.compaction.SizeTieredCompactionStrategy', 'max_threshold': '32', 'min_threshold': '4'}
    AND compression = {'chunk_length_in_kb': '64', 'class': 'org.apache.cassandra.io.compress.LZ4Compressor'}
    AND crc_check_chance = 1.0
    AND dclocal_read_repair_chance = 0.1
    AND default_time_to_live = 0
    AND gc_grace_seconds = 864000
    AND max_index_interval = 2048
    AND memtable_flush_period_in_ms = 0
    AND min_index_interval = 128
    AND read_repair_chance = 0.0
    AND speculative_retry = '99PERCENTILE';
```

```
cqlsh:books_keyspace> select * from books_by_title limit 10;

 book_title                        | authors                     | book_category
-----------------------------------+-----------------------------+----------------
     El amor en tiempos de cólera | {'Gabriel García Márquez'} |          novela
 Las cuatro esquinas del universo |               {'Various'} |         ficcion
                             Mort |        {'Terry Pratchett'} | fantasia epica
                The Fountainhead |              {'Ayn Rand'} |          novela
    The Man in Search of Meaning |          {'Viktor Frankl'} |       filosofia
              El viejo y el mar |       {'Ernest Hemingway'} |          novela
           Las mil y una noches |            {'Desconocido'} |        infantil
             Clockwork princess |       {'Cassandra Claire'} |        fantasia
           La sombra del viento |      {'Carlos Ruiz Zafón'} |          novela
          La cabaña del tio Tom |  {'Harriet Beecher Stowe'} |          novela
```

Figura 8: *books_by_title*

```
cqlsh:books_keyspace> DESCRIBE books_by_username;

CREATE TABLE books_keyspace.books_by_username (
    username text,
    book_title text,
    book_category text,
    book_authors set<text>,
    score smallint,
    PRIMARY KEY (username, book_title, book_category)
) WITH CLUSTERING ORDER BY (book_title ASC, book_category ASC)
    AND bloom_filter_fp_chance = 0.01
    AND caching = {'keys': 'ALL', 'rows_per_partition': 'NONE'}
    AND comment = ''
    AND compaction = {'class': 'org.apache.cassandra.db.compaction.SizeTieredCompactionStrategy', 'max_threshold': '32', 'min_threshold': '4'}
    AND compression = {'chunk_length_in_kb': '64', 'class': 'org.apache.cassandra.io.compress.LZ4Compressor'}
    AND crc_check_chance = 1.0
    AND dclocal_read_repair_chance = 0.1
    AND default_time_to_live = 0
    AND gc_grace_seconds = 864000
    AND max_index_interval = 2048
    AND memtable_flush_period_in_ms = 0
    AND min_index_interval = 128
    AND read_repair_chance = 0.0
    AND speculative_retry = '99PERCENTILE';
```

```
cqlsh:books_keyspace> select * from books_by_username limit 10;

 username | book_title                      | book_category | book_authors           | score
----------+---------------------------------+---------------+------------------------+-------
    David |                         Drácula |        novela |       {'Bram Stoker'} |    10
    David |        El fantasma de la ópera |        novela |     {'Gastón Leroux'} |    10
    David |      El retrato de Dorain Gray |        novela |       {'Oscar Wilde'} |     9
    David |           La historia del loco |        novela |   {'Jhon Katzenbach'} |    10
    David |               Un bien al mundo |        novela |     {'Andrea Bajani'} |    10
  rodfiso |                  Artemis Fowl |      fantasia |       {'Eoin Colfer'} |    10
  rodfiso |            Crónicas marcianas  |       ficcion |      {'Ray Bradbury'} |     9
  rodfiso |            Cuentos de Terramar |      fantasia | {'Ursula K. LeGuinn'} |     9
  rodfiso | El libro de las cosas perdidas |      fantasia |    {'John Connelly '} |     8
  rodfiso |                      Espejismo |       ficcion |        {'Hugh Howey'} |     9
```

Figura 9: *books_by_username*

```
cqlsh:books_keyspace> DESCRIBE books_by_score;

CREATE TABLE books_keyspace.books_by_score (
    score smallint,
    book_title text,
    book_category text,
    authors set<text>,
    PRIMARY KEY (score, book_title, book_category)
) WITH CLUSTERING ORDER BY (book_title ASC, book_category ASC)
    AND bloom_filter_fp_chance = 0.01
    AND caching = {'keys': 'ALL', 'rows_per_partition': 'NONE'}
    AND comment = ''
    AND compaction = {'class': 'org.apache.cassandra.db.compaction.SizeTieredCompactionStrategy', 'max_threshold': '32', 'min_threshold': '4'}
    AND compression = {'chunk_length_in_kb': '64', 'class': 'org.apache.cassandra.io.compress.LZ4Compressor'}
    AND crc_check_chance = 1.0
    AND dclocal_read_repair_chance = 0.1
    AND default_time_to_live = 0
    AND gc_grace_seconds = 864000
    AND max_index_interval = 2048
    AND memtable_flush_period_in_ms = 0
    AND min_index_interval = 128
    AND read_repair_chance = 0.0
    AND speculative_retry = '99PERCENTILE';
```

```
cqlsh:books_keyspace> select * from books_by_score limit 10;

 score | book_title                                                  | book_category  | authors
-------+-------------------------------------------------------------+----------------+---------------------------
     6 |                                  Atrapados en la escuela |        ficcion |               {'Various'}
     6 |                                          Juego de tronos | fantasia epica |    {'George R. R. Martin'}
     3 |                                              Planilandia |        ficcion |    {'Edwin Abbott Abbott'}
    10 |                                  A confederacy of dunces |         novela |     {'John Kennedy Toole'}
    10 |                                             Artemis Fowl |       fantasia |            {'Eoin Colfer'}
    10 |                                            Atlas Shrugged |         novela |              {'Ayn Rand'}
    10 | Bart Simpsons guide to life: A wee handbook for the perplexed |      educacion |               {'Various'}
    10 |                          Calculo integral de varias variables |      educacion |   {'Javier Paéz Cardenas'}
    10 |                                            Choque de reyes | fantasia epica |    {'George R. R. Martin'}
    10 |                                     Cien años de soledad |         novela | {'Gabriel García Márquez'}
```

Figura 10: *books_by_score*

```
cqlsh:books_keyspace> DESCRIBE category_by_name;

CREATE TABLE books_keyspace.category_by_name (
    category text PRIMARY KEY
) WITH bloom_filter_fp_chance = 0.01
    AND caching = {'keys': 'ALL', 'rows_per_partition': 'NONE'}
    AND comment = ''
    AND compaction = {'class': 'org.apache.cassandra.db.compaction.SizeTieredCompactionStrategy', 'max_threshold': '32', 'min_threshold': '4'}
    AND compression = {'chunk_length_in_kb': '64', 'class': 'org.apache.cassandra.io.compress.LZ4Compressor'}
    AND crc_check_chance = 1.0
    AND dclocal_read_repair_chance = 0.1
    AND default_time_to_live = 0
    AND gc_grace_seconds = 864000
    AND max_index_interval = 2048
    AND memtable_flush_period_in_ms = 0
    AND min_index_interval = 128
    AND read_repair_chance = 0.0
    AND speculative_retry = '99PERCENTILE';
```

```
cqlsh:books_keyspace> select * from category_by_name limit 10;

 category
----------------
         poesia
    divulgacion
 fantasia epica
       infantil
          satira
      autoayuda
       fantasia
       politica
         novela
       thriller
```

Figura 11: *category_by_name*

- *queries_keyspace*:

```
cqlsh:queries_keyspace> DESCRIBE favorite_category_by_client;

CREATE TABLE queries_keyspace.favorite_category_by_client (
    username text,
    category_frequency int,
    category text,
    PRIMARY KEY (username, category_frequency)
) WITH CLUSTERING ORDER BY (category_frequency DESC)
    AND bloom_filter_fp_chance = 0.01
    AND caching = {'keys': 'ALL', 'rows_per_partition': 'NONE'}
    AND comment = ''
    AND compaction = {'class': 'org.apache.cassandra.db.compaction.SizeTieredCompactionStrategy', 'max_threshold': '32', 'min_threshold': '4'}
    AND compression = {'chunk_length_in_kb': '64', 'class': 'org.apache.cassandra.io.compress.LZ4Compressor'}
    AND crc_check_chance = 1.0
    AND dclocal_read_repair_chance = 0.1
    AND default_time_to_live = 0
    AND gc_grace_seconds = 864000
    AND max_index_interval = 2048
    AND memtable_flush_period_in_ms = 0
    AND min_index_interval = 128
    AND read_repair_chance = 0.0
    AND speculative_retry = '99PERCENTILE';
```

```
cqlsh:queries_keyspace> select * from favorite_category_by_client limit 10;

 username | category_frequency | category
----------+--------------------+-----------
  rodfiso |                 11 |   ficcion
  rodfiso |                  3 |  fantasia
  rodfiso |                  1 | thriller
  antonio |                  3 |    novela
  antonio |                  1 |  infantil
    bondi |                  3 |  politica
    bondi |                  2 |  historia
    bondi |                  1 |  finanzas
  artemio |                  6 |    novela
  artemio |                  2 | filosofia
```

Figura 12: *favorite_category_by_client*

```
cqlsh:queries_keyspace> DESCRIBE best_books_by_category;

CREATE TABLE queries_keyspace.best_books_by_category (
    book_category text,
    score smallint,
    book_title text,
    authors set<text>,
    PRIMARY KEY ((book_category, score), book_title)
) WITH CLUSTERING ORDER BY (book_title ASC)
    AND bloom_filter_fp_chance = 0.01
    AND caching = {'keys': 'ALL', 'rows_per_partition': 'NONE'}
    AND comment = ''
    AND compaction = {'class': 'org.apache.cassandra.db.compaction.SizeTieredCompactionStrategy', 'max_threshold': '32', 'min_threshold': '4'}
    AND compression = {'chunk_length_in_kb': '64', 'class': 'org.apache.cassandra.io.compress.LZ4Compressor'}
    AND crc_check_chance = 1.0
    AND dclocal_read_repair_chance = 0.1
    AND default_time_to_live = 0
    AND gc_grace_seconds = 864000
    AND max_index_interval = 2048
    AND memtable_flush_period_in_ms = 0
    AND min_index_interval = 128
    AND read_repair_chance = 0.0
    AND speculative_retry = '99PERCENTILE';
```

```
cqlsh:queries_keyspace> select * from best_books_by_category limit 10;

 book_category | score | book_title                                    | authors
---------------+-------+-----------------------------------------------+--------------------------
        novela |    10 |                       A confederacy of dunces |     {'John Kennedy Toole'}
        novela |    10 |                                  Atlas Shrugged |              {'Ayn Rand'}
        novela |    10 |                     Cien años de soledad | {'Gabriel García Márquez'}
        novela |    10 |                                    Circe |        {'Madeline Miller'}
        novela |    10 |                                  Drácula |           {'Bram Stoker'}
        novela |    10 |                       El Conde de Montecristo |        {'Alexandre Dumas'}
        novela |    10 | El abuelo que saltó por la ventana y se largo |         {'Jonas Jonasson'}
        novela |    10 |                   El amor en tiempos de cólera | {'Gabriel García Márquez'}
        novela |    10 |                       El fantasma de la ópera |          {'Gastón Leroux'}
        novela |    10 | El ingenioso hidalgo don Quijote de la Mancha |    {'Miguel de Cervantes'}
```

Figura 13: *best_books_by_category*

```
cqlsh:queries_keyspace> DESCRIBE most_satisfied_clients_by_book;

CREATE TABLE queries_keyspace.most_satisfied_clients_by_book (
    book_title text,
    score smallint,
    username text,
    country text,
    authors set<text>,
    book_category text,
    membership text,
    PRIMARY KEY ((book_title, score), username, country)
) WITH CLUSTERING ORDER BY (username ASC, country ASC)
    AND bloom_filter_fp_chance = 0.01
    AND caching = {'keys': 'ALL', 'rows_per_partition': 'NONE'}
    AND comment = ''
    AND compaction = {'class': 'org.apache.cassandra.db.compaction.SizeTieredCompactionStrategy', 'max_threshold': '32', 'min_threshold': '4'}
    AND compression = {'chunk_length_in_kb': '64', 'class': 'org.apache.cassandra.io.compress.LZ4Compressor'}
    AND crc_check_chance = 1.0
    AND dclocal_read_repair_chance = 0.1
    AND default_time_to_live = 0
    AND gc_grace_seconds = 864000
    AND max_index_interval = 2048
    AND memtable_flush_period_in_ms = 0
    AND min_index_interval = 128
    AND read_repair_chance = 0.0
    AND speculative_retry = '99PERCENTILE';
```

```
cqlsh:queries_keyspace> select * from most_satisfied_clients_by_book limit 10;

 book_title                    | score | username | country        | authors                      | book_category | membership
-------------------------------+-------+----------+----------------+------------------------------+---------------+------------
                      Carry on |     9 |  veleros | AntipodaOscura |         {'Rainbow Rowell'} |         novela |   Diamante
          La princesa de hielo |     8 |  rodfiso |         México |       {'Camilla Lackberg'} |      thriller  |   Estándar
                   Papá Goriot |     9 |  alfonso |         Mexico |       {'Honoré de Balzac'} |         novela |   Estándar
                   Planilandia |     3 |  veleros | AntipodaOscura |    {'Edwin Abbott Abbott'} |        ficcion |   Diamante
                Suite francesa |     9 |  veleros | AntipodaOscura |       {'Irene Nemirovsky'} |      biografia |   Diamante
           Cien años de soledad |    10 |  alfonso |         Mexico | {'Gabriel García Márquez'} |         novela |   Estándar
 El libro de las cosas perdidas |     8 |  rodfiso |         México |         {'John Connelly '} |       fantasia |   Estándar
    El ajedrez de Bobby Fischer |    10 |  alfonso |         Mexico |             {'Elie Agur'} |        ajedrez |   Estándar
            Slaughterhouse five |     8 |   néstor |         México |          {'Kurt Vonnegut'} |         novela |   Estándar
     The making of modern Japan |     8 |   néstor |         México |          {'Marius Jansen'} |       historia |   Estándar
```

Figura 14: *most_satisfied_clients_by_book*

- *clients_keyspace*:

```
cqlsh:clients_keyspace> DESCRIBE clients_by_username;

CREATE TABLE clients_keyspace.clients_by_username (
    username text,
    last_name text,
    country text,
    first_name text,
    membership text,
    PRIMARY KEY (username, last_name, country)
) WITH CLUSTERING ORDER BY (last_name ASC, country ASC)
    AND bloom_filter_fp_chance = 0.01
    AND caching = {'keys': 'ALL', 'rows_per_partition': 'NONE'}
    AND comment = ''
    AND compaction = {'class': 'org.apache.cassandra.db.compaction.SizeTieredCompactionStrategy', 'max_threshold': '32', 'min_threshold': '4'}
    AND compression = {'chunk_length_in_kb': '64', 'class': 'org.apache.cassandra.io.compress.LZ4Compressor'}
    AND crc_check_chance = 1.0
    AND dclocal_read_repair_chance = 0.1
    AND default_time_to_live = 0
    AND gc_grace_seconds = 864000
    AND max_index_interval = 2048
    AND memtable_flush_period_in_ms = 0
    AND min_index_interval = 128
    AND read_repair_chance = 0.0
    AND speculative_retry = '99PERCENTILE';
```

```
cqlsh:clients_keyspace> select * from clients_by_username ;

 username  | last_name | country         | first_name | membership
-----------+-----------+-----------------+------------+------------
   rodfiso |  Figueroa |          México |    Rodolfo |   Estándar
  antonio |    Aguilar |          México |    Antonio |   Estándar
    bondi |    Ramírez |          México |  Alejandro |   Estándar
  artemio |    Padilla |         Nigeria |    Artemio |   Diamante
 patricio |    Barrero |          México |   Patricio |   Estándar
   carlos |  Cerritos |          México |     Carlos |   Diamante
     raul |  Mosqueda |         Viltrum |       Raul |   Diamante
   néstor |  Martínez |          México |     Néstor |   Estándar
   avilix | Hernández |          México |     Avilix |   Estándar
   pamela |      Ruiz |          México |     Pamela |         Oro
 guillermo |      Cota |         Viltrum |  Guillermo |         Oro
     joel |    Avalos |          México |       Joel |   Diamante
     hugo |     Morán |          Mexico |       Hugo |   Diamante
  alfonso |   Barajas |          Mexico |    Alfonso |   Estándar
   israel |   Cabello |          México |     Israel |   Estándar
    david |     Rojas |          México |      David |         Oro
 fernando |  Tiburcio |          México |   Fernando |   Diamante
   cradik |     Yáñez |          Mexico |     Marcos |   Diamante
  veleros |   Veleros | AntipodaOscura |       Luis |   Diamante
   benito |    Franco |         Nigeria |     Benito |   Estándar
```

Figura 15: *clients_by_username*

```
cqlsh:clients_keyspace> DESCRIBE clients_by_country

CREATE TABLE clients_keyspace.clients_by_country (
    country text,
    username text,
    first_name text,
    last_name text,
    membershIp text,
    PRIMARY KEY (country, username)
) WITH CLUSTERING ORDER BY (username ASC)
    AND bloom_filter_fp_chance = 0.01
    AND caching = {'keys': 'ALL', 'rows_per_partition': 'NONE'}
    AND comment = ''
    AND compaction = {'class': 'org.apache.cassandra.db.compaction.SizeTieredCompactionStrategy', 'max_threshold': '32', 'min_threshold': '4'}
    AND compression = {'chunk_length_in_kb': '64', 'class': 'org.apache.cassandra.io.compress.LZ4Compressor'}
    AND crc_check_chance = 1.0
    AND dclocal_read_repair_chance = 0.1
    AND default_time_to_live = 0
    AND gc_grace_seconds = 864000
    AND max_index_interval = 2048
    AND memtable_flush_period_in_ms = 0
    AND min_index_interval = 128
    AND read_repair_chance = 0.0
    AND speculative_retry = '99PERCENTILE';
```

```
cqlsh:clients_keyspace> select * from clients_by_country;

 country        | username  | first_name | last_name | membership
----------------+-----------+------------+-----------+------------
 AntipodaOscura |   veleros |       Luis |   Veleros |   Diamante
        Viltrum | guillermo |   Guillermo |      Cota |         Oro
        Viltrum |      raul |       Raul |  Mosqueda |   Diamante
         Mexico |   alfonso |     Alfonso |   Barajas |   Estándar
         Mexico |    cradik |      Marcos |     Yáñez |   Diamante
         Mexico |      hugo |       Hugo |     Morán |   Diamante
        Nigeria |   artemio |     Artemio |   Padilla |   Diamante
        Nigeria |    benito |      Benito |    Franco |   Estándar
         México |   antonio |     Antonio |    Aguilar |   Estándar
         México |    avilix |      Avilix | Hernández |   Estándar
         México |     bondi |  Alejandro |    Ramírez |   Estándar
         México |    carlos |      Carlos |  Cerritos |   Diamante
         México |     david |       David |     Rojas |         Oro
         México |  fernando |    Fernando |  Tiburcio |   Diamante
         México |    israel |      Israel |   Cabello |   Estándar
         México |      joel |        Joel |    Avalos |   Diamante
         México |    néstor |      Néstor |  Martínez |   Estándar
         México |    pamela |      Pamela |      Ruiz |         Oro
         México |  patricio |    Patricio |   Barrero |   Estándar
         México |   rodfiso |     Rodolfo |  Figueroa |   Estándar
```

Figura 16: *clients_by_country*

```
cqlsh:clients_keyspace> DESCRIBE clients_by_membership

CREATE TABLE clients_keyspace.clients_by_membership (
    membership text,
    username text,
    country text,
    first_name text,
    last_name text,
    PRIMARY KEY (membership, username)
) WITH CLUSTERING ORDER BY (username ASC)
    AND bloom_filter_fp_chance = 0.01
    AND caching = {'keys': 'ALL', 'rows_per_partition': 'NONE'}
    AND comment = ''
    AND compaction = {'class': 'org.apache.cassandra.db.compaction.SizeTieredCompactionStrategy', 'max_threshold': '32', 'min_threshold': '4'}
    AND compression = {'chunk_length_in_kb': '64', 'class': 'org.apache.cassandra.io.compress.LZ4Compressor'}
    AND crc_check_chance = 1.0
    AND dclocal_read_repair_chance = 0.1
    AND default_time_to_live = 0
    AND gc_grace_seconds = 864000
    AND max_index_interval = 2048
    AND memtable_flush_period_in_ms = 0
    AND min_index_interval = 128
    AND read_repair_chance = 0.0
    AND speculative_retry = '99PERCENTILE';
```

```
cqlsh:clients_keyspace> select * from clients_by_membership ;

 membership | username  | country        | first_name | last_name
------------+-----------+----------------+------------+-----------
   Estándar |   alfonso |         Mexico |     Alfonso |   Barajas
   Estándar |   antonio |         México |     Antonio |   Aguilar
   Estándar |    avilix |         México |      Avilix | Hernández
   Estándar |    benito |        Nigeria |      Benito |    Franco
   Estándar |     bondi |         México |  Alejandro |   Ramírez
   Estándar |    israel |         México |      Israel |   Cabello
   Estándar |    néstor |         México |      Néstor |  Martínez
   Estándar |  patricio |         México |    Patricio |   Barrero
   Estándar |   rodfiso |         México |     Rodolfo |  Figueroa
   Diamante |   artemio |        Nigeria |     Artemio |   Padilla
   Diamante |    carlos |         México |      Carlos |  Cerritos
   Diamante |    cradik |         México |      Marcos |     Yáñez
   Diamante |  fernando |         México |    Fernando |  Tiburcio
   Diamante |      hugo |         Mexico |       Hugo |     Morán
   Diamante |      joel |         México |       Joel |    Avalos
   Diamante |      raul |        Viltrum |       Raul |  Mosqueda
   Diamante |   veleros | AntipodaOscura |       Luis |   Veleros
        Oro |     david |         México |      David |     Rojas
        Oro | guillermo |        Viltrum |  Guillermo |      Cota
        Oro |    pamela |         México |     Pamela |      Ruiz
```

Figura 17: *clients_by_membership*

2.5. Verificación de consultas

- **Q3**: Obtener los mejores libros de una categoría dada

```
1 SELECT * FROM best_books_by_category WHERE score IN (9,10) AND
2     book_category = 'politica' LIMIT 10;
```

```
cqlsh:queries_keyspace> SELECT * FROM best_books_by_category WHERE score IN (9,10) AND
                    ...     book_category = 'politica' LIMIT 10;

 book_category | score | book_title                                                        | authors
---------------+-------+-------------------------------------------------------------------+-----------------------
      politica |     9 |                                          ¡Sálvese quien pueda! | {'Andrés Oppenheimer'}
      politica |    10 |                                            How Democracies Die |    {'Steven Levitsky'}
      politica |    10 |                                                      República |            {'Platón'}
      politica |    10 |                                           Twitter and Tear Gas |     {'Zeynep Tufekci'}
      politica |    10 | Why nations fail: The origins of power, prosperity and poverty |     {'Daren Acemoglu'}
```

Figura 18: Resultados de la query **Q3**

- **Q8**: Obtener la categoría preferida de un cliente dado

```
SELECT username, category, category_frequency AS frequency FROM
    favorite_category_by_client WHERE username = 'n stor' GROUP BY username;
```

```
cqlsh:queries_keyspace> SELECT username, category, category_frequency AS frequency FROM
                    ...     favorite_category_by_client WHERE username = 'néstor' GROUP BY username;

 username | category | frequency
----------+----------+-----------
   néstor |   novela |         5
```

Figura 19: Resultados de la query **Q8**

- **Q9**: Obtener los clientes que más disfrutaron un libro

```
SELECT * FROM most_satisfied_clients_by_book WHERE score IN (8, 9, 10) AND
    book_title = 'Cien a os de soledad';
```

```
cqlsh:queries_keyspace> SELECT * FROM most_satisfied_clients_by_book WHERE score IN (8, 9, 10) AND
                    ...     book_title = 'Cien años de soledad';

 book_title           | score | username  | country | authors                      | book_category | membership
----------------------+-------+-----------+---------+------------------------------+---------------+------------
 Cien años de soledad |     9 | guillermo | Viltrum | {'Gabriel García Márquez'} |       novela |        Oro
 Cien años de soledad |    10 |   alfonso |  Mexico | {'Gabriel García Márquez'} |       novela |   Estándar
```

Figura 20: Resultados de la query **Q9**

3. Conclusiones

Si bien las consultas descritas e implementadas son relativamente sencillas para la persona que las está realizando, la mayor desventaja que yo encuentro con el enfoque de diseñar orientado a consultas es que gran parte de la inserción de datos se tiene que realizar a nivel de aplicación así como el aseguramiento de la consistencia de la información dentro de la base de datos.

Es justo en este punto donde me parece relevante realizar una análisis de cuando podemos implementar Cassandra como la base de datos principal dentro de un negocio. En mi opinión, y dicho tanto por Lakshman como por Malik ([2]), si la consistencia no es relevante para nuestro giro de negocio (como en la búsqueda de mensajes dentro de la app de Messenger de Facebook) Cassandra es ideal por la velocidad y escalabilidad flexible. Sin embargo, si nos interesa un gran nivel de consistencia, aunque podamos modificarla, Cassandra no es una gran idea a menos que se especifiquen reglas muy estrictas a nivel de aplicación sobre la inserción de información. Hago énfasis en la inserción porque con un enfoque orientado a consultas, el principal problema es que hay información idéntica que se tiene que insertar en tablas diferentes y si se olvida hacerla en alguna de ellas, existirá inconsistencia. Por ejemplo, la información en las tablas `clients_by_username`, `clients_by_country` y `clients_by_membership` la información es la misma, lo único que cambia

es la *partition key* lo cual facilita y acelera mucho el proceso de consulta dentro de un sector de la aplicación.

Referencias

[1] Apache. (2021). Apache Cassandra Documentation v4.0-rc1. Obtenido en https://cassandra.apache.org/doc/latest/ el 8 de mayo del 2021.

[2] Lakshman A., & Malik P. (2009). Cassandra - A Decentralized Structured Storage System. Obtenido de la Universidad de Cornell el 8 de mayo del 2021.

[3] Carpenter J., & Hewitt E. (2020). Cassandra: The Definitive Guide. Third Ed. O'Reilly Media.

[4] Hobbs T. Basic Rules of Cassandra Data Modeling. (2015). DataStax. Obtenido en https://www.datastax.com/blog/basic-rules-cassandra-data-modeling el 10 de mayo del 2021.

[5] Lerer B. A deep look at the CQL WHERE clause. (2015). DataStax. Obtenido en https://www.datastax.com/blog/deep-look-cql-where-clause el 11 de mayo del 2021.