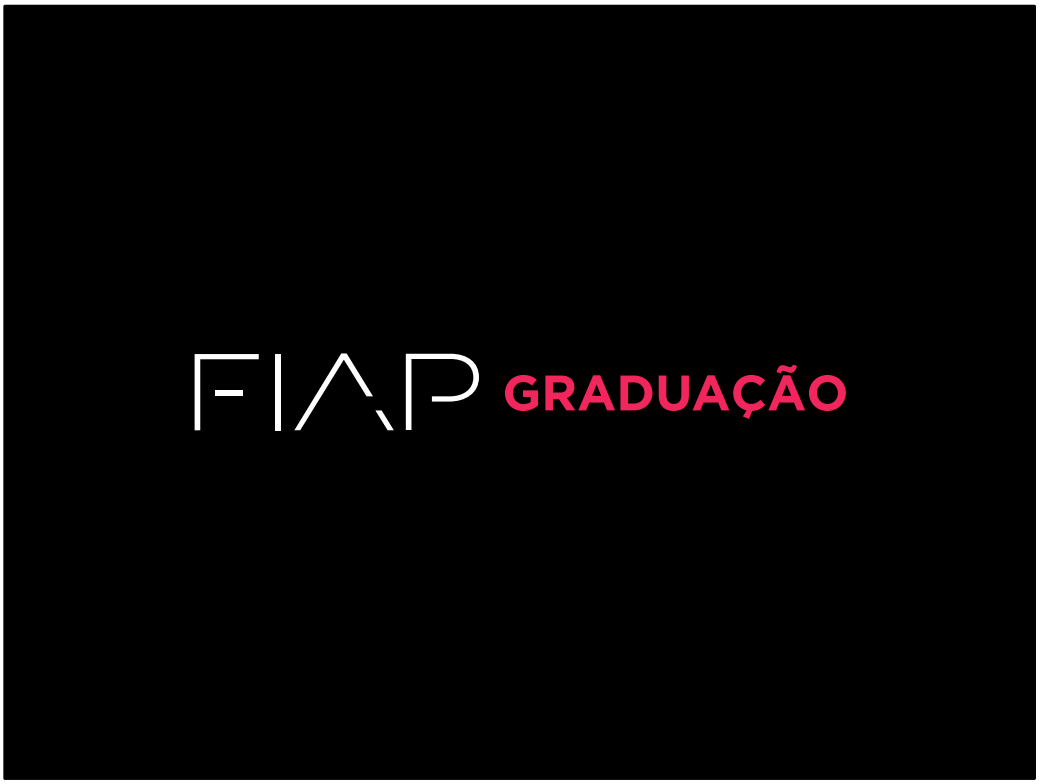
FIAP
GRADUAÇÃO

FIAP

Architecture Analytics & NoSQL

NoSQL Databases – Column Store - Cassandra

Milton

Versão 1.2 - fevereiro de 24

FIAP

NoSQL
CQL

3

CQL

FIAP

- Lista de comandos disponíveis

```
HELP;
```

```
token@cqlsh:escola> help;

Shell command help topics:
========================
CAPTURE  CLS          COPY     EXIT    HELP   PAGING  SHOW    TIMING   UNICODE
CLEAR    CONSISTENCY  EXECUTE  EXPAND  LOGIN  SERIAL  SOURCE  TRACING

Full documentation for shell commands:
  https://docs.datastax.com/en/dse/6.8/cql/cql/cql_reference/cqlsh_commands/cqlshCommandsTOC.html

Full documentation for CQL commands:
  https://docs.datastax.com/en/dse/6.8/cql/cql/cql_reference/cql_commands/cqlCommandsTOC.html
```

CQL

FIAP

- O comando DESCRIBE CLUSTER no CQL (Cassandra Query Language) é usado para obter informações sobre o cluster Cassandra no qual você está conectado. Esse comando fornece detalhes sobre a topologia do cluster, incluindo informações sobre os nós (ou servidores) que compõem o cluster e outras informações de configuração.

DESCRIBE CLUSTER;

```
token@cqlsh:escola> describe cluster;

Cluster: cndb
Partitioner: Murmur3Partitioner

Range ownership:
                    -9060123382903608438  [10.182.144.26, 10.16.60.3, 10.16.62.6]
                    -8249275723965299284  [10.16.60.3, 10.182.144.26, 10.16.62.6]
                    -7585753150739611567  [10.182.144.26, 10.16.62.6, 10.16.60.3]
                    -7185881258711174824  [10.16.62.6, 10.16.60.3, 10.182.144.26]
                    -6986422665871156355  [10.16.60.3, 10.16.62.6, 10.182.144.26]
                    -6215487520884968675  [10.16.62.6, 10.182.144.26, 10.16.60.3]
                    -4811032350624723228  [10.182.144.26, 10.16.60.3, 10.16.62.6]
                    -4702777212499211885  [10.16.60.3, 10.16.62.6, 10.182.144.26]
                    -4335011188634626570  [10.16.62.6, 10.182.144.26, 10.16.60.3]
                    -3826695241688554548  [10.182.144.26, 10.16.60.3, 10.16.62.6]
                    -3776507746142668516  [10.16.60.3, 10.182.144.26, 10.16.62.6]
                    -2631025927865771841  [10.182.144.26, 10.16.60.3, 10.16.62.6]
                    -1855545014770939566  [10.16.60.3, 10.182.144.26, 10.16.62.6]
                    -1543592404949266258  [10.182.144.26, 10.16.62.6, 10.16.60.3]
                      849137741115454115  [10.16.62.6, 10.182.144.26, 10.16.60.3]
                     1346671748220170523  [10.182.144.26, 10.16.62.6, 10.16.60.3]
                     2382932542363344697  [10.16.62.6, 10.182.144.26, 10.16.60.3]
                     2388327187567989511  [10.16.62.6, 10.182.144.26, 10.16.60.3]
                     3153436341962145122  [10.182.144.26, 10.16.60.3, 10.16.62.6]
                     3161003850705231874  [10.16.60.3, 10.16.62.6, 10.182.144.26]
                     3507407185662258033  [10.16.62.6, 10.16.60.3, 10.182.144.26]
                     4681681598289715633  [10.16.60.3, 10.16.62.6, 10.182.144.26]
                     5775025211491891095  [10.16.62.6, 10.16.60.3, 10.182.144.26]
                     9122778818678718950  [10.16.60.3, 10.182.144.26, 10.16.62.6]
```

5

CQL

FIAP

- O comando DESCRIBE KEYSPACES é usado para listar informações sobre todos os keyspaces disponíveis no cluster Cassandra ao qual você está conectado. Ele fornece uma visão geral dos keyspaces presentes no cluster, incluindo seus nomes e as configurações de replicação associadas a cada um deles.

DESCRIBE KEYSPACES;

```
token@cqlsh:escola> DESCRIBE KEYSPACES;

escola              datastax_sla   system_traces
data_endpoint_auth  system_schema  system_virtual_schema
system_auth         system         system_views
```

6

CQL

FIAP

- O comando SHOW VERSION é usado para exibir a versão do software do Cassandra que está atualmente em execução no nó ou nó de destino. Este comando pode ser útil para verificar rapidamente a versão do Cassandra instalada e em uso em um nó específico.

```
SHOW VERSION;
```

CQL

FIAP

- O comando PAGING é útil para controlar como os resultados das consultas são exibidos no console CQL, especialmente quando você está lidando com grandes conjuntos de dados e deseja controlar a quantidade de resultados exibidos de cada vez.

```
PAGING;
```

CQL

FIAP

- A opção CONSISTENCY permite controlar o nível de consistência dos dados lidos ou gravados em um cluster Cassandra.
- Existem vários níveis de consistência disponíveis no Cassandra, incluindo:
  - ONE: Isso significa que apenas um nó no cluster precisa confirmar a operação. Isso é mais rápido, mas menos consistente.
  - QUORUM: A maioria dos nós (mais da metade) no cluster deve confirmar a operação. Isso oferece um bom equilíbrio entre consistência e disponibilidade.
  - ALL: Todos os nós no cluster devem confirmar a operação. Isso oferece a maior consistência, mas pode ser mais lento e pode exigir que todos os nós estejam disponíveis.
  - LOCAL_QUORUM: A maioria dos nós em um data center local deve confirmar a operação.
  - EACH_QUORUM: A maioria dos nós em cada data center (local e remoto) deve confirmar a operação.
  - ANY: Qualquer nó no cluster pode confirmar a operação. Isso é o mais rápido, mas o menos consistente.

CONSISTENCY;

O QUE É UM KEYSPACE

FIAP

- **Keyspace é um Conceito Central:** Um keyspace é uma estrutura fundamental no Cassandra para organizar dados relacionados.
  - **Agrupa Tabelas e Dados:** Serve como um contêiner que agrupa tabelas e suas famílias de colunas associadas.
  - **Configuração de Replicação:** Cada keyspace permite configurar como os dados são replicados entre os nós do cluster.
  - **Estratégia de Replicação:** Define como os dados são distribuídos; Estratégias comuns incluem SimpleStrategy e NetworkTopologyStrategy.
  - **Namespace e Isolamento:** Evita conflitos de nomes entre tabelas e fornece isolamento lógico para as tabelas.
  - **Controle de Acesso:** Permite a aplicação de permissões de controle de acesso para garantir a segurança dos dados.
  - **Políticas de Expiração e Durabilidade:** Controla quanto tempo os dados são retidos e como são armazenados.
  - **Unidade Organizacional Essencial:** É fundamental para o modelo de dados distribuído e altamente escalável do Cassandra.
  - **Flexibilidade e Escalabilidade:** O uso de keyspaces permite flexibilidade na organização de dados e escalabilidade horizontal.
  - **Gerenciamento de Dados Eficiente:** Facilita o gerenciamento eficiente de dados em um ambiente distribuído.

CQL

FIAP

- CREATE KEYSPACE: Usado para criar um namespace de keyspace, que é o equivalente a um banco de dados em Cassandra.

```
CREATE KEYSPACE scott_keyspace
  WITH replication = {'class': 'SimpleStrategy',
'replication_factor': 1};

DESCRIBE KEYSPACES;

USE scott_keyspace;
```

CQL

FIAP

- Para a criação de COLUMN FAMILIES, use a sintaxe simples:

```
CREATE TABLE IF NOT EXISTS
keyspace_name.table_name
( column_definition,
column_definition, ...)
WITH property AND property ...
```

column_name cql_type **STATIC PRIMARY KEY**

|column_name <tuple<tuple_type> tuple<tuple_type>... > PRIMARY KEY

|column_name frozen<user-defined_type> PRIMARY KEY

|column_name frozen<collection_name><collection_type>... PRIMARY KEY

|PRIMARY KEY (partition_key)

CQL

FIAP

- Exemplo da criação de uma tabela com valores ordenados

```
CREATE TABLE timeseries (
  event_type text,
  insertion_time timestamp,
  event blob,
  PRIMARY KEY (event_type, insertion_time) )
WITH CLUSTERING ORDER BY (insertion_time
DESC);
```

13

CQL

FIAP

- uso de compactação

```
CREATE COLUMNFAMILY sblocks (
  block_id uuid,
  subblock_id uuid,
  data blob,
  PRIMARY KEY (block_id,
subblock_id) )
WITH COMPACT STORAGE;
```

CQL

FIAP

- Podemos especificar propriedades que afetam o comportamento dos dados que residirão na tabela

```
CREATE TABLE monkeySpecies (
  block_id uuid,
  species text,
  average_size text,
  population varint,
  PRIMARY KEY (species, block_id) )
WITH caching = { 'keys' : 'NONE', 'rows_per_partition' :
'120' };
```

CQL

FIAP

- Exercícios
- Crie a tabela abaixo:

```
CREATE TABLE user (
 first_name text,
 last_name text,
 PRIMARY KEY (first_name));


DESCRIBE TABLE user;
```

16

CQL

FIAP

- Exercícios:
- Crie uma tabela com 3 colunas, ID (integer), name (texto) e email (texto). Defina a primary key para coluna ID. Chame essa tabela de **users**.
- Crie uma tabela com 4 colunas, posted_on (big integer), user_id (integer), user_name (texto), body (texto - corpo da mensagem) e defina a primary key composta pelas colunas user_id e posted_on. Chame essa tabela de **messages**.
- Descreva ambas as tabelas criadas.

17

CQL

FIAP

- Assim como nos KEYSPACES, podemos alterar algumas propriedades e estruturas das tabelas

```
ALTER TABLE monkeySpecies ALTER
average_size TYPE varint;

ALTER TABLE monkeySpecies ADD gravesite
varchar;

ALTER TABLE user ADD title text;

DESCRIBE TABLE user;
```

CQL

FIAP

- O comando TRUNCATE é usado para eliminar todos os dados de uma tabela

```
TRUNCATE monkeySpecies;
```

- O comando DROP é usado para eliminar os dados e a estrutura de uma tabela.

```
DROP TABLE monkeySpecies;
```

CQL

FIAP

- Exercícios
- Adicione uma nova coluna chamada password (texto) na tabela users.
- Adicione outra coluna chamada password_reset_token (texto) na tabela users.
- Descreva a tabela users.

CQL

FIAP

- Insira os seguintes dados na tabela users

```
INSERT INTO users (id, name, email) VALUES (101, 'otto', 'otto@abc.de');
INSERT INTO users (id, name) VALUES (102, 'jane');
INSERT INTO users (id, name) VALUES (103, 'karl');
INSERT INTO users (id, name, email) VALUES (104, 'linda', 'linda@abc.de');
INSERT INTO users (id, name, email) VALUES (105, 'gerd', 'g@rd.de');
INSERT INTO users (id, name, email) VALUES (106, 'heinz','heinz@xyz.de');
```

CQL

FIAP

- Insira os seguintes dados na tabela messages

```
INSERT INTO messages (user_id, posted_on, user_name, body) VALUES (101, 1384895178,
'otto', 'Hello World!');
INSERT INTO messages (user_id, posted_on, user_name, body) VALUES (101, 1384895319,
'otto', 'Hello again...');
INSERT INTO messages (user_id, posted_on, user_name, body) VALUES (104, 1384895222,
'linda', 'Hi, Otto');
INSERT INTO messages (user_id, posted_on, user_name, body) VALUES (103, 1384895223,
'karl', 'Politic sucks');
INSERT INTO messages (user_id, posted_on, user_name, body) VALUES (103, 1384895224,
'karl', 'rhabarber rhabarber rhabarber');
INSERT INTO messages (user_id, posted_on, user_name, body) VALUES (103, 1384895225,
'karl', 'want desperate eat a burger');
INSERT INTO messages (user_id, posted_on, user_name, body) VALUES (103, 1384895226,
'karl', 'Avengers Infinite War has lots of problems about quantic phisics concepts');
INSERT INTO messages (user_id, posted_on, user_name, body) VALUES (103, 1384895227,
'karl', 'A pale shelter');
```

CQL

FIAP

- Insira a seguinte linha na tabela user

```
INSERT INTO user (first_name, last_name)
VALUES ('Bill', 'Nguyen’);

SELECT COUNT (*) FROM user;
```

CQL

FIAP

- Um SET (conjunto) consiste em um grupo de elementos com valores únicos. Valores duplicados não serão armazenados de forma distinta. Os valores de um conjunto são armazenados de forma não ordenada, mas retornarão os elementos na ordem de classificação quando consultados.

```
CREATE TABLE images (
name text PRIMARY KEY,
owner text,
date timestamp,
tags set<text>) ;

INSERT INTO images (name, owner, tags)
   VALUES ('cat.jpg', 'jsmith', { 'pet',
'cute' });
```

CQL

FIAP

- Uma lista (LIST) tem uma forma semelhante a um SET, em que uma lista agrupa e armazena valores. Ao contrário de um SET, os valores armazenados em uma lista não precisam ser exclusivos e podem ser duplicados. Além disso, uma lista armazena os elementos em uma ordem específica e pode ser inserida ou recuperada de acordo com um valor de índice.

```
CREATE TABLE plays (
id text PRIMARY KEY,
game text,
players int,
scores list<int>) ;

INSERT INTO plays (id, game, players, scores)
VALUES ('123-afde', 'quake', 3, [17, 4, 2]);
```

CQL

FIAP

- Crie a tabela EMP

```
DROP TABLE EMP;

CREATE TABLE emp(
   emp_id int,
   emp_name text,
   emp_city text,
   emp_sal varint,
   emp_phone varint,
   primary key ((emp_city), emp_id)
   );
```

CQL

FIAP

- Insira os dados na tabela EMP

```
INSERT INTO emp (emp_id, emp_name, emp_city, emp_phone, emp_sal) VALUES(1,'Maria', 'São Paulo',
9848022338, 50000);
INSERT INTO emp (emp_id, emp_name, emp_city,emp_phone, emp_sal) VALUES(2,'Mario', 'Campinas',
9848022339, 40000);
INSERT INTO emp (emp_id, emp_name, emp_city, emp_phone, emp_sal) VALUES(3,'Ana', 'Nova Odessa',
9848022330, 45000);
INSERT INTO EMP (EMP_ID, EMP_CITY, EMP_NAME, EMP_PHONE, EMP_SAL) VALUES (4, 'Campinas', 'Carla',
9123456789, 6000);
INSERT INTO EMP (EMP_ID, EMP_CITY, EMP_NAME, EMP_PHONE, EMP_SAL) VALUES (5, 'Campinas', 'Tanya',
9123456788, 7000);
INSERT INTO EMP (EMP_ID, EMP_CITY, EMP_NAME, EMP_PHONE, EMP_SAL) VALUES (6, 'Campinas', 'Kirito',
9123456787, 8000);
INSERT INTO EMP (EMP_ID, EMP_CITY, EMP_NAME, EMP_PHONE, EMP_SAL) VALUES (7, 'São Paulo', 'Naruto',
9123456786, 7000);
INSERT INTO EMP (EMP_ID, EMP_CITY, EMP_NAME, EMP_PHONE, EMP_SAL) VALUES (8, 'São Paulo', 'Hinata',
9123456786, 10000);
INSERT INTO emp (emp_id, emp_name, emp_city, emp_phone, emp_sal) VALUES(9,'Anita', 'Nova Odessa',
9848022399, 46000);
INSERT INTO emp (emp_id, emp_name, emp_city, emp_phone, emp_sal) VALUES(10,'Akita', 'Nova Odessa',
9848022398, 49000);
```

CQL

FIAP

- Distinct
- Retorna os valore únicos para uma partition key.

```
SELECT EMP_CITY FROM EMP;

SELECT DISTINCT EMP_CITY FROM EMP;
```

28

CQL

FIAP

- LIMIT
- Limita o número máximo retornado por uma consulta

```
SELECT * FROM EMP LIMIT 3;
```

CQL

FIAP

- WHERE
- Permite criar um filtro para os dados

```
SELECT * FROM EMP
 WHERE EMP_CITY = 'Campinas';
```

CQL

FIAP

- WHERE
- Permite criar um filtro para os dados

```
SELECT * FROM EMP
 WHERE EMP_CITY = 'Campinas' and
       EMP_ID > 4;
```

CQL

FIAP

- A instrução GROUP BY em CQL é usada para agrupar linhas que têm os mesmos valores em colunas especificadas.

Função	Descrição
COUNT	Conta o número de linhas na seleção.
SUM	Soma os valores de uma coluna numérica.
AVG	Calcula a média dos valores de uma coluna numérica.
MIN	Encontra o menor valor de uma coluna.
MAX	Encontra o maior valor de uma coluna.

CQL

FIAP

- O Cassandra armazena dados em várias partições, e cada partição é identificada por uma chave de partição. Para realizar uma operação GROUP BY, a chave de partição deve ser especificada na cláusula WHERE da consulta, e a cláusula GROUP BY deve começar com a chave de partição.

33

CQL

FIAP

- Exemplo

```
CREATE TABLE events (
  event_type text,
  event_date date,
  event_id uuid,
  data text,
  PRIMARY KEY (event_type,
event_date, event_id)
);;
```

CQL

FIAP

```
INSERT INTO events (event_type, event_date, event_id, data)
VALUES ('Birthday', '2023-09-07', uuid(), 'Birthday Party');
INSERT INTO events (event_type, event_date, event_id, data)
VALUES ('Birthday', '2023-09-08', uuid(), 'Another Birthday
Party');
INSERT INTO events (event_type, event_date, event_id, data)
VALUES ('Wedding', '2023-10-15', uuid(), 'Wedding Celebration');
INSERT INTO events (event_type, event_date, event_id, data)
VALUES ('Conference', '2023-11-20', uuid(), 'Tech Conference');
INSERT INTO events (event_type, event_date, event_id, data)
VALUES ('Birthday', '2023-12-01', uuid(), 'Yet Another Birthday
Party');
INSERT INTO events (event_type, event_date, event_id, data)
VALUES ('Conference', '2023-11-21', uuid(), 'Another Tech
Conference');
```

CQL

FIAP

```
SELECT * FROM events WHERE event_type =
'Birthday';

SELECT event_type, event_date, COUNT(*)
FROM events
WHERE event_type = 'Birthday'
GROUP BY event_type, event_date;
```

CQL

FIAP

```
SELECT event_type, MIN(event_date) AS oldest, MAX(event_date)
AS newest
FROM events
WHERE event_type IN ('Birthday', 'Wedding', 'Conference')
GROUP BY event_type;
```

CQL

FIAP

- Funções Definidas pelo Usuário (UDFs, do inglês "User-Defined Functions") podem ser usadas para realizar cálculos que não são facilmente realizáveis usando apenas as funções embutidas do CQL.
- As UDFs são executadas no lado do servidor e podem ser invocadas dentro de suas consultas CQL.

```
CREATE OR REPLACE FUNCTION square (input
double)
RETURNS NULL ON NULL INPUT
RETURNS double
LANGUAGE java AS '
  return Double.valueOf(Math.pow(input, 2));
';
```

CQL

FIAP

- Depois de criar uma UDF, você pode usá-la em suas consultas CQL, assim como você usaria qualquer outra função embutida.

```
SELECT emp_id, square(emp_id)
  FROM emp;
```

CQL

FIAP

- A instrução CREATE INDEX em Cassandra é usada para criar um índice secundário em uma ou mais colunas da tabela. Índices secundários são úteis para acelerar consultas em colunas que não fazem parte da chave primária, permitindo que você realize consultas eficientes baseadas em diferentes colunas.

```
CREATE INDEX [IF NOT EXISTS] [index_name]
ON table_name (column_name);
```

CQL

FIAP

- Exemplo de função usando um laço for

```
CREATE OR REPLACE FUNCTION factorial(n
int)
RETURNS NULL ON NULL INPUT
RETURNS int
LANGUAGE java AS '
  int result = 1;
  for (int i = 1; i <= n; i++) {
    result *= i;
  }
  return result;
';
;
```

41

CQL

FIAP

- Exemplo de função usando if-else

```
CREATE OR REPLACE FUNCTION max(a
int, b int)
RETURNS NULL ON NULL INPUT
RETURNS int
LANGUAGE java AS '
  if (a >= b) {
    return a;
  } else {
    return b;
  }
';
```

CQL

FIAP

```
CREATE OR REPLACE FUNCTION instr(input text, character text)
RETURNS NULL ON NULL INPUT
RETURNS int
LANGUAGE java AS '
  if (character == null || character.length() != 1) {
    return -1; // retorna -1 se o caractere for null ou não
tiver exatamente um caractere
  }
  return input.indexOf(character);
';
```

CQL

FIAP

```
CREATE OR REPLACE FUNCTION substr(input text, start_pos int, length int)
RETURNS NULL ON NULL INPUT
RETURNS text
LANGUAGE java AS '
  if (input == null || start_pos < 1 || length < 1) {
    return null;  // retorna null para entradas inválidas
  }

  int adjustedStartPos = start_pos - 1;  // Ajustar para base 0, pois Java usa indexação
baseada em zero
  int endPos = Math.min(adjustedStartPos + length, input.length());  // Não exceder o
comprimento da string de entrada

  if (adjustedStartPos >= input.length()) {
    return "";  // retorna string vazia se a posição inicial estiver fora da string
  }

  return input.substring(adjustedStartPos, endPos);
';
```

FIAP

Referências Bibliográficas

- tutorialspoint. Cassandra Tutorial. Disponível em: http://www.tutorialspoint.com/cassandra/index.htm. Acesso em: 08 agosto 2023.

- DivConq MFT. Introduction to Cassandra Columns, Super Columns and Rows. Disponível em: http://www.divconq.com/2010/cassandra-columns-and-supercolumns-and-rows/. Acesso em: 08 agosto 2023.

- HEWITT, Eben. Cassandra: The Definitive Guide. O'Reilly Media, Inc., 2010. Disponível em: <https://www.safaribooksonline.com/library/view/cassandra-the-definitive/9781449399764/>. Acesso em: 08 agosto 2023.