**Laboratório 1 — NoSQL Cassandra**

https://anderson-paulucci.medium.com/laborat%C3%B3rio-1-nosql-cassandra-629edf27f85b

*Este é um Lab para o desenvolvimento da habilidade de NoSQL Cassandra, de uma sequência de vários outros publicados na plataforma Nano Learning (*[*https://app.nlpro.com.br*](https://app.nlpro.com.br/)*)*

Neste laboratório vamos partir do zero criando um ambiente para o Cassandra na solução Astra da Datastax, Cloud native e isso nos dará flexibilidade para iniciar.

Iremos criar um cenário com modelo de dados hipotético do Twitter.

Acesse o endereço <https://astra.datastax.com/>

1. Crie o seu ambiente com um novo database para usarmos no laboratório:



1. Em alguns minutos estará disponível, acesse o console do CQL. Vamo usar esta interface para facilitar o aprendizado.

**Você está conectado a um cluster Cassandra em funcionamento! Simples né, a versão Astra facilita bastante com o modelo as a service.**



1. Faça o login com o usuário e execute o comando abaixo para listar as keyspaces disponíveis;

describe keyspaces;

Acesse o Keyspace lab01;

use lab01

1. Em NoSQL, a maneira que os seus dados são estruturados esta diretamente ligada como serão recuperados.

CREATE TABLE users(username text PRIMARY KEY,password text);

1. Inserir alguns registros users

INSERT INTO users (username, password) VALUES (‘beto’,’xxxx’);

INSERT INTO users (username, password) VALUES (‘joao’,’yyyy’);

INSERT INTO users (username, password) VALUES (‘beatriz’,’wwww’);

INSERT INTO users (username, password) VALUES (‘ana’,’zzzz’);

1. Todos os amigos e todos os seguidores de um usuário serão armazenados de forma contígua no disco

CREATE TABLE friends (

 username text,

 friend text,

 since timestamp,

 PRIMARY KEY (username, friend)

);

1. Inserir alguns registros friends

INSERT INTO friends (username, friend, since) VALUES (‘beto’,’ana’,1267413962580791);

INSERT INTO friends (username, friend, since) VALUES (‘beto’,’joao’,1267413990076949);

INSERT INTO friends (username, friend, since) VALUES (‘beatriz’,’ana’,1267414008133277);

1. Fazendo uma consulta(muito eficiente) para procurar todos os amigos ou seguidores de um usuário;

CREATE TABLE followers (

 username text,

 follower text,

 since timestamp,

 PRIMARY KEY (username, follower)

);

1. Inserir registros na tabela followers

INSERT INTO followers (username, follower, since) VALUES (‘beatriz’,’ana’,1267414008133277);

INSERT INTO followers (username, follower, since) VALUES (‘beto’,’ana’,1267413962580791);

INSERT INTO followers (username, follower, since) VALUES (‘joao’,’beto’,1267413990076949);

1. Tweets são armazenados com um UUID para a chave:

(obs: Um identificador exclusivo universal (UUID) é um padrão de identificador usado no desenvolvimento de software. O UUID é simplesmente um valor de 128 bits)

CREATE TABLE tweets (

 tweet\_id uuid PRIMARY KEY,

 username text,

 body text

);

1. Inserir alguns registro de tweets

INSERT INTO tweets (tweet\_id, username, body) VALUES (7561a442–24e2–11df-8924–001ff3591711,’ana’,’Ola Pessoal’);

INSERT INTO tweets (tweet\_id, username, body) VALUES (7561a442–24e2–11df-8924–001ff3591712,’ana’,’Iniciando os estudos’);

INSERT INTO tweets (tweet\_id, username, body) VALUES (7561a442–24e2–11df-8924–001ff3591713,’beto’,’aprendendo NoSQL’);

INSERT INTO tweets (tweet\_id, username, body) VALUES (7561a442–24e2–11df-8924–001ff3591714,’beto’,’Na NLPRO’);

1. A ‘userline’ acompanha os tweets em que ordem. Para conseguir isso, usamos o TimeUUID para a chave de cluster,

Dessa forma os tweets serão armazenados em ordem cronológica. A opção “WITH CLUSTERING ORDER BY” define que os tweets serão armazenados em ordem cronológica inversa (primeiro o mais novo), que é um pouco mais eficiente para as consultas que estaremos realizando.

CREATE TABLE userline (

 username text,

 time timeuuid,

 tweet\_id uuid,

 PRIMARY KEY (username, time)

) WITH CLUSTERING ORDER BY (time DESC);

1. Insere dados na tabela userline

INSERT INTO userline (username, time, tweet\_id) VALUES (‘ana’,now(),7561a442–24e2–11df-8924–001ff3591711);

INSERT INTO userline (username, time, tweet\_id) VALUES (‘ana’,now(),7561a442–24e2–11df-8924–001ff3591712);

INSERT INTO userline (username, time, tweet\_id) VALUES (‘beto’,now(),7561a442–24e2–11df-8924–001ff3591713);

INSERT INTO userline (username, time, tweet\_id) VALUES (‘beto’,now(),7561a442–24e2–11df-8924–001ff3591714);

**Laboratório 2 — Exemplos básicos usando o CQL (Cassandra)**

https://anderson-paulucci.medium.com/laborat%C3%B3rio-cassandra-exemplos-b%C3%A1sicos-usando-o-cql-a2512cfa930f

*Este é um Lab para o desenvolvimento da habilidade de NoSQL Cassandra, de uma sequência de vários outros publicados na plataforma Nano Learning (*[*https://app.nlpro.com.br*](https://app.nlpro.com.br/)*)*

1. Criando uma nova tabela;

CREATE TABLE songs (
 id uuid PRIMARY KEY,
 title text,
 album text,
 artist text,
 data blob
 );

CREATE TABLE playlists (
 id uuid,
 song\_order int,
 song\_id uuid,
 title text,
 album text,
 artist text,
 PRIMARY KEY (id, song\_order ) );

1. Executando o insert na tabela playlists;

INSERT INTO playlists (id, song\_order, song\_id, title, artist, album)
 VALUES (62c36092–82a1–3a00–93d1–46196ee77204, 4,
 7db1a490–5878–11e2-bcfd-0800200c9a66,
 ‘Ojo Rojo’, ‘Fu Manchu’, ‘No One Rides for Free’);

1. Inserindo UUID usando a função now();

INSERT INTO playlists (id, song\_order, song\_id, title, artist, album)

VALUES

(now(), 4, 7db1a490–5878–11e2-bcfd-0800200c9a66, ‘Ojo Rojo’, ‘Fu Manchu’, ‘No One Rides for Free’);

1. Executando filtro na coluna da tabelas;

*Obs: vai dar erro na execução, sempre é recomendável fazer a consulta usando a chave.*

SELECT album, title FROM playlists WHERE artist = ‘Fu Manchu’;

1. Criar um index secundário;

CREATE INDEX ON playlists( artist );

1. Execute novamente a consulta;

SELECT album, title FROM playlists WHERE artist = ‘Fu Manchu’;

1. O CQL contém várias estruturas que podem ser interessantes no modelo, alguns tipos de coleção:

• Conjunto
• Lista
• Mapa

Em um banco de dados relacional, para permitir que os usuários tenham vários endereços de e-mail, você cria uma tabela de endereço de e-mail com uma relação de muitos-para-um (join) para uma tabela de usuários. O CQL lida com o caso de uso do endereço de e-mail múltiplo clássico e outros casos de uso, definindo colunas como coleções. Usar o tipo de coleção definida para resolver o problema de endereços de e-mail múltiplo é conveniente e intuitivo.

CREATE TABLE users ( user\_id text PRIMARY KEY, first\_name text, last\_name text, emails set<text>);

1. Inserindo dados na coleção;

INSERT INTO users (user\_id, first\_name, last\_name, emails) VALUES(‘frodo’, ‘Frodo’, ‘Baggins’, {‘f@baggins.com’, ‘baggins@gmail.com’});

1. Selecionando todas as coleções;

SELECT user\_id, emails FROM users WHERE user\_id = ‘frodo’;

1. Adicionando dados na coleção;

UPDATE users SET emails = emails + {‘fb@friendsofmordor.org’} WHERE user\_id = ‘frodo’;

1. Removendo dados da coleção;

UPDATE users SET emails = emails — {‘fb@friendsofmordor.org’} WHERE user\_id = ‘frodo’;

1. Atualizando uma coleção para adicionar tags na tabela;

ALTER TABLE playlists ADD tags set<text>;

1. Executando o update para atualização;

UPDATE playlists SET tags = tags + {‘2007’} WHERE id = 62c36092–82a1–3a00–93d1–46196ee77204 AND song\_order = 2;

UPDATE playlists SET tags = tags + {‘covers’} WHERE id = 62c36092–82a1–3a00–93d1–46196ee77204 AND song\_order = 2;

UPDATE playlists SET tags = tags + {‘1973’} WHERE id = 62c36092–82a1–3a00–93d1–46196ee77204 AND song\_order = 1;

UPDATE playlists SET tags = tags + {‘blues’} WHERE id = 62c36092–82a1–3a00–93d1–46196ee77204 AND song\_order = 1;

UPDATE playlists SET tags = tags + {‘rock’} WHERE id = 62c36092–82a1–3a00–93d1–46196ee77204 AND song\_order = 4;

**Laboratório 3 — Modelo de IoT usando o Cassandra**

https://anderson-paulucci.medium.com/laborat%C3%B3rio-3-modelo-de-iot-usando-o-cassandra-9af942601094

Vamos considerar uma estação meteorológica, criando dados de temperatura a cada minuto.

Você vai entender como usar Cassandra como um poderosa ferramenta para persistência de dados de IoT, com uma modelagem de dados adequada para este contexto de séries temporais.

**Séries Temporais**: Cassandra é impressionante em séries temporais

1. IoT com persistência no Cassandra:



1. Criar uma nova tabela temperatura;

CREATE TABLE temperatura (weatherstation\_id text,event\_time timestamp,temperatura text,PRIMARY KEY (weatherstation\_id,event\_time));

1. Inserir dados na tabela;

INSERT INTO temperatura(weatherstation\_id, event\_time, temperatura) VALUES (‘1234ABCD’,’2019–04–03 07:05:00',’10');

INSERT INTO temperatura(weatherstation\_id, event\_time, temperatura) VALUES (‘1234ABCD’,’2019–04–03 07:06:00',’10');

INSERT INTO temperatura(weatherstation\_id, event\_time, temperatura) VALUES (‘1234ABCD’,’2019–04–03 07:07:00',’11');

INSERT INTO temperatura(weatherstation\_id, event\_time, temperatura) VALUES (‘1234ABCD’,’2019–04–03 07:08:00',’11');

INSERT INTO temperatura(weatherstation\_id, event\_time, temperatura) VALUES (‘1234ABCD’,’2019–04–03 07:09:00',’12');

INSERT INTO temperatura(weatherstation\_id, event\_time, temperatura) VALUES (‘1234ABCD’,’2019–04–03 07:10:00',’12');

INSERT INTO temperatura(weatherstation\_id, event\_time, temperatura) VALUES (‘1234ABCD’,’2019–04–03 07:11:00',’12');

INSERT INTO temperatura(weatherstation\_id, event\_time, temperatura) VALUES (‘1234ABCD’,’2019–04–03 07:12:00',’13');

INSERT INTO temperatura(weatherstation\_id, event\_time, temperatura) VALUES (‘1234ABCD’,’2019–04–03 07:13:00',’13');

INSERT INTO temperatura(weatherstation\_id, event\_time, temperatura) VALUES (‘1234ABCD’,’2019–04–03 07:14:00',’13');

INSERT INTO temperatura(weatherstation\_id, event\_time, temperatura) VALUES (‘1234ABCD’,’2019–04–03 07:15:00',’13');

INSERT INTO temperatura(weatherstation\_id, event\_time, temperatura) VALUES (‘1234ABCD’,’2019–04–03 07:16:00',’12');

INSERT INTO temperatura(weatherstation\_id, event\_time, temperatura) VALUES (‘1234ABCD’,’2019–04–03 07:17:00',’12');

INSERT INTO temperatura(weatherstation\_id, event\_time, temperatura) VALUES (‘1234ABCD’,’2019–04–03 07:18:00',’12');

INSERT INTO temperatura(weatherstation\_id, event\_time, temperatura) VALUES (‘1234ABCD’,’2019–04–03 07:19:00',’11');

INSERT INTO temperatura(weatherstation\_id, event\_time, temperatura) VALUES (‘1234ABCD’,’2019–04–03 07:20:00',’10');

INSERT INTO temperatura(weatherstation\_id, event\_time, temperatura) VALUES (‘1234ABCD’,’2019–04–03 07:21:00',’10');

INSERT INTO temperatura(weatherstation\_id, event\_time, temperatura) VALUES (‘1234ABCD’,’2019–04–03 07:22:00',’10');

INSERT INTO temperatura(weatherstation\_id, event\_time, temperatura) VALUES (‘1234ABCD’,’2019–04–03 07:23:00',’09');

INSERT INTO temperatura(weatherstation\_id, event\_time, temperatura) VALUES (‘1234ABCD’,’2019–04–03 07:24:00',’09');

1. Consultar tabela usando filtro;

SELECT event\_time, temperature

FROM temperature

WHERE weatherstation\_id=’1234ABCD’;

1. Consultar com filtro por range de dados, \*usando a chave;

SELECT temperature

FROM temperature

WHERE weatherstation\_id=’1234ABCD’

AND event\_time > ‘2019–04–03 07:01:00’;

1. Séries Temporais — Limitar o tamanho da linha (“partição” de linha);

CREATE TABLE temperatura\_dia (

 weatherstation\_id text,

 date text,

 event\_time timestamp,

 temperatura text,

 PRIMARY KEY ((weatherstation\_id,date),event\_time));

1. Inserir dados na nova tabela;

INSERT INTO

 temperatura\_dia(weatherstation\_id, date, event\_time, temperatura)

 VALUES (‘1234ABCD’,’2019–04–03',’2013–04–03 07:01:00',’72F’);

INSERT INTO

 temperatura\_dia(weatherstation\_id, date, event\_time, temperatura)

 VALUES (‘1234ABCD’,’2019–04–03',’2019–04–03 07:02:00',’73F’);

INSERT INTO

 temperatura\_dia(weatherstation\_id, date, event\_time, temperatura)

 VALUES (‘1234ABCD’,’2019–04–04',’2019–04–04 07:01:00',’73F’);

INSERT INTO

 temperatura\_dia(weatherstation\_id, date,event\_time, temperatura)

 VALUES (‘1234ABCD’,’2019–04–04',’2019–04–04 07:02:00',’74F’);

1. Consultar por data;

SELECT \*

FROM temperatura\_dia

WHERE weatherstation\_id=’1234ABCD’

AND date=’2019–04–03';

1. Com Cassandra, podemos tirar proveito de um recurso chamado expirando colunas e expurgar os dados após um determinado período de segundos.

CREATE TABLE temperatura\_atual (

 weatherstation\_id text,

 event\_time timestamp,

 temperatura text,

 PRIMARY KEY (weatherstation\_id,event\_time),

) WITH CLUSTERING ORDER BY (event\_time DESC);

1. Inserindo os dados. Observe o TTL de 20 que significa que os dados irá expirar em 20 segundos.

INSERT INTO

 temperatura\_atual(weatherstation\_id, event\_time, temperatura)

 VALUES (‘1234ABCD’,’2019–04–03 07:03:00',’72F’) USING TTL 20;

INSERT INTO

 temperatura\_atual(weatherstation\_id, event\_time, temperatura)

 VALUES (‘1234ABCD’,’2019–04–03 07:02:00',’73F’) USING TTL 20;

INSERT INTO

 temperatura\_atual(weatherstation\_id, event\_time, temperatura)

 VALUES (‘1234ABCD’,’2019–04–03 07:01:00',’73F’) USING TTL 20;

INSERT INTO

 temperatura\_atual(weatherstation\_id, event\_time, temperatura)

 VALUES (‘1234ABCD’,’2019–04–03 07:04:00',’74F’) USING TTL 20;