Introdução

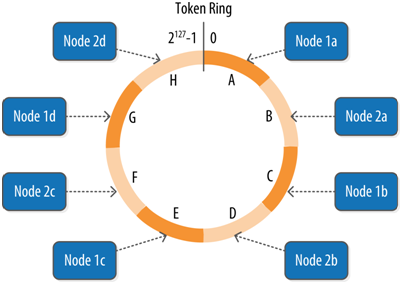
O Cassandra foi inicialmente desenvolvido pelo Facebook para ser utilizado no motor de busca de sua caixa de entrada de mensagens. Em 2008 ele se tornou *open-source* e em 2009 passou a ser mantido pela Apache Foundation. Seu modelo de distribuição do sistema é baseado no Dynamo (desenvolvido pela Amazon).

Arquitetura

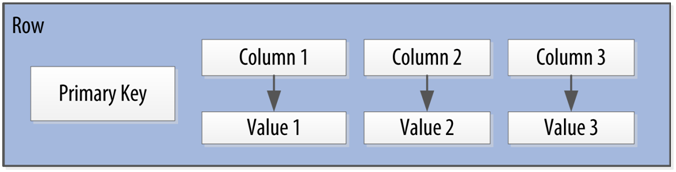
Por ter uma arquitetura distribuída e descentralizada, uma solução com Cassandra é altamente escalável, principalmente para escalar de forma horizontal. Caso seja necessária mais performance, basta adicionar mais nós na rede, não é necessário *a priori* substituir as máquinas existentes por servidores mais poderosos. Da mesma forma, se a demanda diminuir, pode-se remover alguns nós e economizar na manutenção do sistema.

Os dados das tabelas do Cassandra passam automaticamente por um processo de particionamento utilizando a *partition key* da tabela que, por padrão, é baseada na chave primária da tabela. As partições são então distribuídas entre os nós do cluster utilizando uma técnica conhecida como *data sharding*. As partições são replicadas em múltiplos nós para prover alta disponibilidade e tolerância a falhas, o número de réplicas é configurado na definição de um *keyspace*.

Para determinar em que nó do cluster cada partição será gravada, são definidas faixas baseadas no *hash* da chave utilizada para particionamento, essas faixas são chamadas de *tokens* e são representadas por um inteiro de 64 bits. Cada nó assume responsabilidade por um ou mais *tokens* até cobrirem todas as faixas possíveis. Esse tipo de arquitetura é conhecido como Arquitetura em Anel, representado pela figura abaixo:

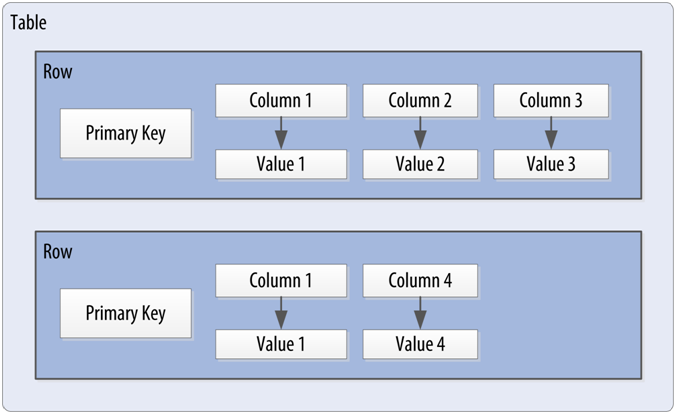


Cassandra é um banco NoSQL do tipo chave-valor. Cada conjunto de chave-valor é equivalente ao que seria uma coluna e um conjunto de colunas associadas a uma chave primária compõem o que seria uma linha de uma tabela do Cassandra, como representado na figura abaixo:



Estrutura de uma linha de uma tabela no Cassandra

Diferente de uma tabela de um banco relacional, no Cassandra não é necessário popular todas as colunas ao criar uma nova linha, de forma que podemos ter “larguras” diferentes para cada uma delas (uma coluna de um banco relacional sempre precisa ser preenchida com algum valor, mesmo que seja um valor nulo). A figura abaixo representa esse conceito.



Representação de uma tabela do Cassandra

Ao gravar um valor em uma coluna, junto a ele é associado um *timestamp* indicando o horário exato em que o valor foi gravado. Isso é importante para detectar versões desatualizadas de valores quando o Cassandra comparar o resultado de uma consulta feita em vários nós diferentes.

### 

### **A linguagem CQL**

A linguagem CQL *(Cassandra Query Language)* é utilizada para interagir com o Cassandra. A vantagem de se trabalhar com o CQL é que ele é próximo do mais tradicional SQL, sendo familiar para quem vem do mundo dos bancos relacionais.

Os comandos no CQL são divididos em três categorias básicas, sendo comandos para:

* Definição das estruturas de dados;
* Manipulação dos dados;
* Consultas.

### **Criando um banco de dados no Cassandra**

##### 

**create\_keyspace\_statement** ::= CREATE KEYSPACE [ IF NOT EXISTS ]**nome\_keyspace** WITH [**options**](https://cassandra.apache.org/doc/latest/cql/ddl.html#grammar-token-options)

**CREATE** **KEYSPACE** excelsior

**WITH** replication **=** {'class': 'SimpleStrategy', 'replication\_factor' : 3};

**CREATE** **KEYSPACE** excalibur

**WITH** replication **=** {'class': 'NetworkTopologyStrategy', 'DC1' : 1, 'DC2' : 3}

**AND** durable\_writes **=** false;

use\_statement ::= USE **nome\_keyspace**

**CREATE TABLE monkeySpecies (**

**species text PRIMARY KEY,**

**common\_name text,**

**population varint,**

**average\_size int**

**) WITH comment='Important biological records';**

**CREATE TABLE timeline (**

**userid uuid,**

**posted\_month int,**

**posted\_time uuid,**

**body text,**

**posted\_by text,**

**PRIMARY KEY (userid, posted\_month, posted\_time)**

**) WITH compaction = { 'class' : 'LeveledCompactionStrategy' };**

**CREATE TABLE loads (**

**machine inet,**

**cpu int,**

**mtime timeuuid,**

**load float,**

**PRIMARY KEY ((machine, cpu), mtime)**

**) WITH CLUSTERING ORDER BY (mtime DESC);**

A cláusula WHERE especifica quais linhas devem ser consultadas. É composto pelas relações nas colunas que fazem parte do e / ou possuem um índice secundário definido nas mesmas.PRIMARY KEY