Banco de Dados NoSQL

Aula 7: Modelo de dados NoSQL baseado em colunas – parte I

Apresentação

Já estudamos, por meio de exemplos,o projeto de um banco de dados orientado a documentos com o uso de modelos de agregados. Agora, estudaremos o modelo de banco de dadosbaseado em colunas usando o DataStax, que é a mais conhecida distribuição do Apache Cassandra.

Objetivo

- Descrever as características do banco de dados baseados em colunas;
- Analisar as vantagens do banco de dados colunar em relação ao modelo relacional;
- Escrever comandos NOSQL para a definição da estrutura do banco de dados.

Introdução

Após verificarmos os modelos de dados chave-valor e orientado a documentos, conheceremos o modelo baseado em colunas (SADALAGE; FOWLER, 2013), também conhecido como colunar. Utilizaremos como sistema gerenciador de banco de dados (SGBD) o Cassandra, mais especificamente em sua distribuição Datastax.

O Cassandra é, segundo o ranking apresentado no DB-Engines, o mais usado dos SGBDs baseados em colunas, seguido do HBase e do Microsoft Azure Cosmos DB, o Google BigTable e o Hypertable. Esse tipo de modelo de banco de dados NOSQL é usado por grandes empresas, como Apple, Instagram, McDonald's, Netflix, Github, Spotify, Uber e Walmart, entre muitas outras.

Saiba mais

Segundo dados da página oficial do Cassandra, somente a Apple armazena no Cassandra aproximadamente 10 Petabytes de dados distribuídos em 75.000 nós, ou seja, é um banco de dados usado para suporte de grandes volumes de dados.

Primeiro, detalharemos as características do banco de dados baseados em colunas; depois,analisaremos as vantagens do banco colunar em relação ao modelo relacional; e, por fim, escreveremos comandos NOSQL para a definição da estrutura nesse modelo.

Características do banco de dados baseados em colunas

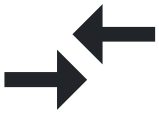
O Apache Cassandra é um banco de dados distribuído, descentralizado e do tipo software livre. Foi criado não apenas para o armazenamento de grandes volumes de dados, mas para o processamento de grandes quantidades de dados, já quea existência de aplicações envolvendo grandes quantidades é algo comum na área de bancos de dados.

Porém, o rápido acesso foi uma necessidade que explodiu diante do uso massivo de dados com a Internet, conhecido como Big Data. As redes sociais,entre outros tipos de aplicações, provocaram uma procura por banco de dados coma produção de respostas rápidas aos usuários.

Assim, o Cassandra foi criado pelo próprio Facebook para atender às suas necessidades, sendo que depois a empresa disponibilizou o código-fonte para a comunidade, em 2008.

Modelo colunar

A principal característica do modelo colunar é que os dados são organizados em colunas e uma família de colunas corresponde a uma coleção de linhas semelhantes. O modelo de dados do Cassandra usa o conceito de Keyspace, que é um contêiner ou um agrupamento de dados.



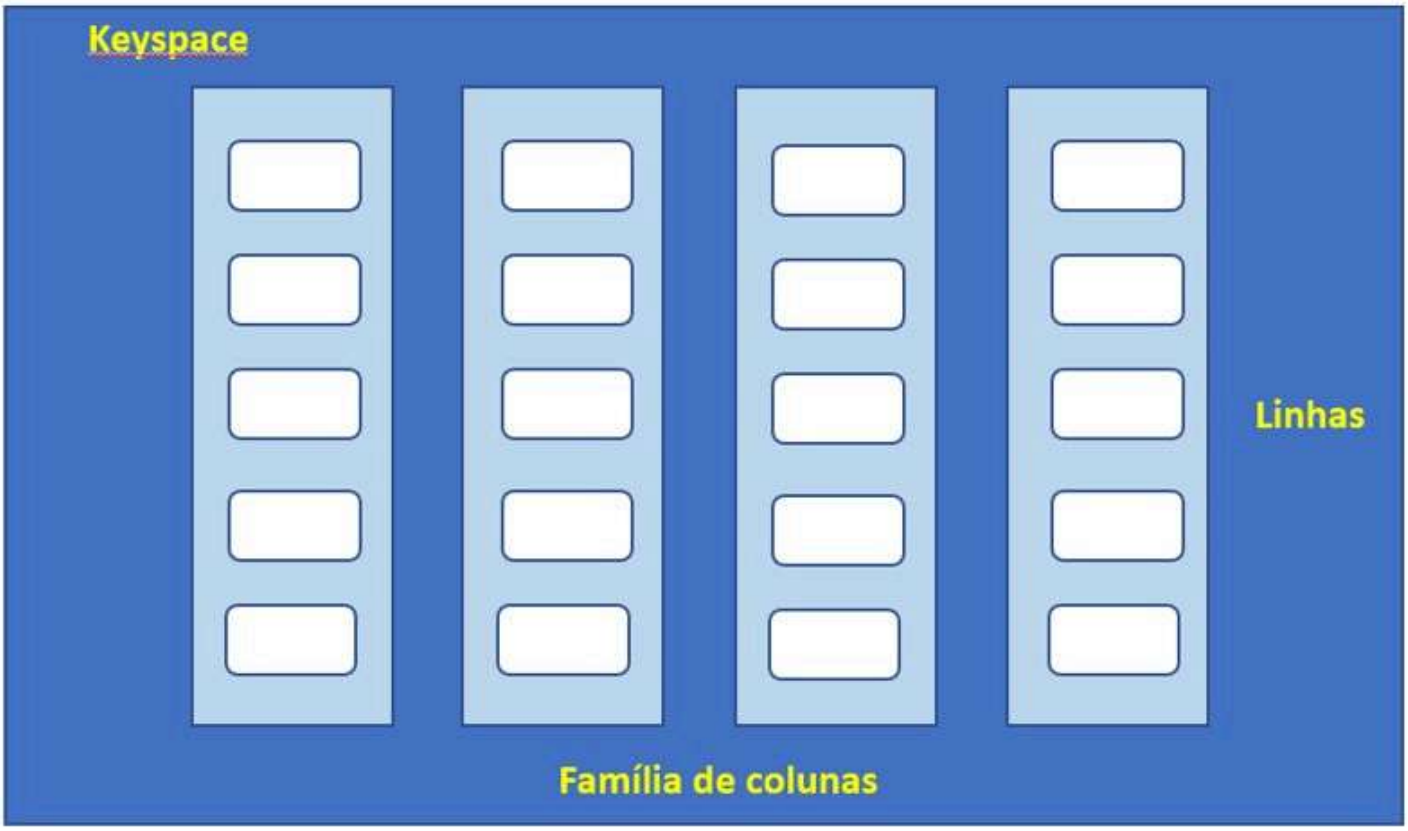
Modelo relacional

No modelo relacional, isso corresponde a um banco de dados formado por um esquema (ou *schema*), que é um conjunto de tabelas. Lembrando que, em SGBDs relacionais como o PostgreSQL, quando não é definido, o banco de dados cria o esquema *public* para a organização das tabelas.

No contêiner, uma linha no banco de dados é uma coleção de colunas comuma associação entre elas por uma determinada chave. Essas colunas são armazenadas com base em seus nomes e o nome tem como objetivo a identificação das linhas, ou

seja, são usadas como a chave para a pesquisa dos registros.

Figura 1 - Modelo de dados usado no Cassandra



Fonte: Apache Cassandra.

Ainda em relação às características do modelo colunar e, mais especificamente no caso do Cassandra, o banco de dados apresenta como características principais:

Clique nos botões para ver as informações.

Característica 1

É próprio para aplicações descentralizadas, sendo ele formado por um conjunto de nós. Com isso, obtém-se mais segurança, pois se elimina os pontos únicos de falha em um sistema distribuído, em quedeve serconsiderado que os dados são replicados pelos nós, para que seja possível fornecer serviços de banco de dados com alta disponibilidade.

Característica 2

É linearmente escalável, o que na prática apresenta como consequência um aumento da taxa de transferência, de acordo com o aumento do número de nós de um cluster. Além disso, o modelo possui também outro ponto positivo, que é permitir a execução de consultas paralelas.

Vantagens do banco de dados colunar em relação ao modelo relacional

Ao contrário do modelo relacional, o modelo orientado a colunas não possui relacionamentos, ou seja, não tem suporte ao uso de chaves estrangeiras e, com isso, não implementa a integridade referencial. Os bancos de dados relacionais são orientados ou organizados em linhas (registros de uma tabela), pois os dados em cada linha de uma tabela são armazenados juntos, por atributos ou campos de uma tabela.

No banco de dados colunar, diferentemente da forma usada na estrutura do banco de dados relacional, os dados são organizados e armazenados em colunas. Essa diferença, embora aparente ser muito simples, é uma característica determinante dos bancos de dados colunares.

O Cassandra não faz uso de transações, como nos bancos de dados relacionais, baseadas nas propriedades ACID (Atomicidade-Consistência-Isolamento-Durabilidade) (ELMASRI; NAVATHE, 2018).

Comentário

Essas propriedades permitem a aplicação de mecanismos de recuperação ou bloqueio de transações em blocos de comandos, bem como que as transações sejam executadas de forma atômica, isoladas umas das outras e com persistência após a conclusão das operações. Porém, essa é uma escolha e não uma ausência de recurso!

O banco de dados colunar não foi projetado para uso em sistemas que necessitam de controle transacional com as propriedades ACID ou para o uso com consultas envolvendo agregações (operações do tipo *groupby*), como o uso da função de somatório,tendo em vista a própria estrutura do banco de dados.

Os modelos orientados a colunas, como é característica dos bancos de dados NOSQL, usam o Teorema CAP (do Inglês *Consistency – Availability–Partition tolerance*) para o controle de consistência.

Quanto às terminologias usadas no modelo relacional e no orientado a colunas, temos as seguintes equivalências:

1

Uma instância de bancos de dados no relacional corresponde a um cluster no colunar.y

2

A *database/schema* corresponde a um keyspace.

3

Uma tabela corresponde a uma família de colunas.

4

Uma linha é uma linha em ambos; uma coluna no relacional tem os mesmos tipos de dados e é igual para todas as linhas;já no colunar, as colunas podem ser diferentes ao longo das linhas.

Comandos NOSQL para a definição da estrutura do banco de dados

Agora conheceremos, na prática, o modelo baseado em colunas. Para isso, utilizaremos o SGBD Cassandra no Windows e a distribuição Datastaxem sua versão Community, sem usar a interface gráfica, ou seja, trabalharemos em linha de comando com o *Cassandra Query Language Shell* (CQLSH) para ter um foco maior nos comandos e não nos recursos da interface.

Uma das características marcantes dos bancos NOSQL é que eles não são limitados ao uso da SQL, ou seja, possuem uma linguagem própria para a manipulação de dados. A linguagem do Cassandra é a CQL – *Cassandra Query Language*, que suporta comandos do tipo SQL próprios para o modelo colunar.

Primeiro, temos que criar um keyspace, que é um contêiner de dados no Cassandra. Ele desempenha o mesmo papel de um database em SGBD relacional. Crie um *keyspace* em um cluster de avaliação de nó único:

```
CREATE KEYSPACE Aula

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

No comando, há a indicação da quantidade de réplicas, bem como a determinação de quantas cópias dos dados são mantidas

em um determinado *datacenter*. Essa configuração necessita de um estudo mais aprofundado, pois afeta a consistência da aplicação, a disponibilidade e a performance como um todo.

Em nossa introdução ao banco de dados colunar no Cassandra, utilizaremos a configuração padrão indicada para avaliação do banco de dados e para ambientes de desenvolvimento e de teste, considerando um *datacenter* único.

Depois de criado o *keyspace*, podemos usar o comando abaixo:

```
DESCRIBE keyspaces;
```

Com o comando Use, podemos identificar o *keyspace* a ser usado na sessão atual pelo cliente. Depois de definido o *keyspace*, todos os comandos seguintesaplicados nas tabelas serão realizados no ambiente desse *keyspace*, a não ser que seja especificado outro *keyspace* ou até o momento do encerramento da conexão do cliente. Para tal, usar o comando (observe o uso de ponto e vírgula no fim de cada comando):

```
USE aula;
```

Podemos verificar a estrutura do *keyspace* criado pelo comando apresentando a estrutura logo abaixo do comando:

```
DESCRIBE aula ;

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

A propriedade durable_writes = true significa que os comandos de manipulação de dados escreverão no *commit log* antes de escreverem no *keyspace*, garantindo a durabilidade (persistência) do resultado. O valor default é *true*, que é mandatório no caso de class = SimpleStrategy.

Caso necessário, pode-se destruir um *keyspace* com o comando:

Atenção! Aqui existe uma videoaula, acesso pelo conteúdo online

```
DROP keyspaceaula;
```

Comentário

Existe a possibilidade de alterar as propriedades de um keyspace com o comando ALTER KEYSPACE seguindo a sintaxe:
ALTER KEYSPACE <keyspace name> WITH <properties> AND <properties> ;

Antes de continuarmos a verificar os detalhes dos comandos da linguagem de definição de dados no Cassandra, vamos conhecer os tipos de dados que podem ser usados no banco de dados.

Tipos de dados

Clique no botão acima.

Os tipos estão ordenados alfabeticamente:

Nome do tipo de dado	Tipo de dado	Descrição
ascii	Cadeias de caracteres	Usado para a representação de uma cadeia de caracteres do tipo ASCII
bigint	inteiro longo	Indicado para uso com inteiros longos, utiliza 64-bits para armazenamento
Blob	Blobs	Usado para binary large objects
boolean	Booleanos	Usado para valores verdadeiro ou falso
counter	Inteiros	Usado para representar uma coluna numerada
date	Cadeias de caracteres	As datas são cadeias de caracteres seguindo o modelo (yyyy-mm-dd)
decimal	inteiros, decimais	Usado para a representação de números decimais de precisão variável
double	Decimais	Usado para a representação de números decimais de precisão variável com 64 bits
float	Decimais	Usado para a representação de números decimais de precisão variável com 32 bits
Inet	Cadeias de caracteres	Usado para a representação de um endereço IP, IPv4 ou IPv6
Int	Inteiros	Indicado para uso com inteiros, utiliza 32-bits para armazenamento
smallint	Inteiros	Indicado para uso com inteiros curtos, utiliza 16-bits ou 2 bytes para armazenamento
Text	Cadeias de caracteres	Usado para a representação de cadeias de caracteres na codificação UTF8
time	Cadeias de caracteres	Os valores são usados para a representação de números inteiros de 64 bits com a precisão de nanossegundos. Os valores são representados como cadeias de caracteres, como por exemplo 14:40:20,123
timestamp	Cadeias de caracteres	Usado para a representação de um campo do tipo data/hora
timeuuid	UUID	O Cassandra possui várias funções para o uso de dados do tipo timestamp representados como UUID
tinyint	Inteiros	Indicado para uso com inteiros pequenos, utiliza 8-bits ou 1 byte para armazenamento

Uuid	UUID	Universally Unique Identifier
varchar	Cadeias de caracteres	Usado para a representação de um campo do tipo cadeia de caracteres na codificação UTF8
varint	Inteiros	Usado para a representação de um campo do tipo inteiro sem a definição do tamanho
List	Cadeias de caracteres	Uma lista é uma coleção de um ou mais elementos com uma ordenação definida
Map	Cadeias de caracteres	Um mapaéuma coleçãode pares do tipo chave-valor
Set	Cadeias de caracteres	Um conjunto é uma coleção de um ou vários elementos que formam um conjunto (set)

Fonte: Autor.

Atenção! Para visualização completa da tabela utilize a rolagem horizontal

Além dos tipos de dados apresentados, os usuários ainda têm a facilidade de criar seus próprios tipos de dados. A seguir são apresentados os comandos que podem ser usados para a criação e alteração dos tipos de dados definidos pelo usuário. São eles:

1

CREATE TYPE

Usado para a criação de um tipo de dados definido pelo usuário..

2

ALTER TYPE

Usado para a alteração de um tipo de dados definido pelo usuário.

3

DROP TYPE

Usado para a exclusão de um tipo de dados definido pelo usuário.

4

DESCRIBE TYPE

Usado para a descrição de um tipo de dado específico dos definidos pelo usuário.

5

DESCRIBE TYPES

Usado para a descrição dos tipos de dados definidos pelo usuário.

Depois de definido o *keyspace* e do conhecimento dos tipos de dados possíveis de serem usados no Cassandra, verificaremos exemplos de comandos para criar uma família de colunas no banco de dados, sendo essa a mais importante unidade de armazenamento usada no banco de dados. Observe que a CQL é semelhante à SQL:

```
CREATE TABLE cliente (

codigouuid primary key ,

nomevarchar ,

cidadevarchar ) ;
```

Convém ressaltar que, no Cassandra, uma família de colunas é o mesmo que uma tabela, assim o comando poderia ter sido escrito da seguinte maneira:

```
CREATE COLUMNFAMILY cliente2 (

codigouuid primary key ,

nomevarchar ,

cidadevarchar ) ;
```

Atenção! Aqui existe uma videoaula, acesso pelo conteúdo online

Aproveitaremos esse segundo código que criou a família de colunas para apresentar o comando para excluir uma tabela do *keyspace*; esse comando é semelhante ao usado na SQL do banco de dados relacional:

```
DROP COLUMNFAMILY cliente2 ;
```

Para apagar uma tabela, usaremos o mesmo comando da SQL; no caso, não precisamos executar, pois vamos usar a tabela cliente nos exemplos:

```
DROP TABLE cliente ;
```

No caso de alguma alteração ser necessária na estrutura da tabela, devemos usar o comando ALTER TABLE. Vejamos o exemplo para acrescentar um atributo do tipo map na tabela cliente:

```
ALTER TABLE cliente ADD fone map< text, text> ;
```

Para incluir um registro na tabela cliente, podemos usar o comando que contém a função uuid(). Execute os dois comandos INSERT a seguir:

```
INSERT INTO cliente(codigo, nome, cidade, fone)

VALUES (uuid(), 'Ana', 'Rio de Janeiro', {'21' : '1111-2222'});

INSERT INTO cliente(codigo, nome, cidade, fone)

VALUES (uuid(), 'Lia', 'Minas Gerais', {'21' : '1111-2222' , '31' : '2222-3333'});
```

Com um SELECT simples, podemos ver o resultado:

```
-------------------------------------+---------------+----------------------------------------+------

309c705e-cc18-434f-88ee-a1a208f43f1f | Minas Gerais | {'21': '1111-2222', '31': '2222-3333'} |Lia

36720507-be52-4a64-9158-1b85c4e3115b | Rio de Janeiro |{'21': '1111-2222'} |Ana
```

Atenção

O valor do campo código (tipo UUID) é aleatório e pode ser diferente do criado pelo seu banco.

O código a seguir serve para fazer uma atualização da tabela cliente. No caso, será retirado um par chave-valor do atributo fone, que é do tipo map::

```
UPDATE cliente SET fone = fone - {'21'}

WHERE codigo =309c705e-cc18-434f-88ee-a1a208f43f1f ;
```

Outro atributo composto usado no Cassandra é o tipo SET. No exemplo a seguir, será alterada a tabela cliente e acrescentado um atributo para o armazenamento dos contatos de um cliente. Logo, será realizada a atualização dos valores:

```
ALTER TABLE cliente ADD contatos set<text> ;

UPDATE cliente SET contatos = { 'Rui', 'Nei' }

WHERE codigo =309c705e-cc18-434f-88ee-a1a208f43f1f;
```

Atenção! Aqui existe uma videoaula, acesso pelo conteúdo online

Atividade

1. Assinale a afirmativa errada quanto às características observadas em um banco de dados orientado a colunas.

a) O modelo orientado a colunas não tem suporte para a integridade referencial.
b) Ser próprio para aplicações descentralizadas, sendo ele formado por um conjunto de nós.
c) É linearmente escalável, o que na prática apresenta como consequência um aumento da taxa de transferência de acordo com o aumento do número de nós de um *cluster*.
d) As colunas são armazenadas com base em seus nomes e o nome tem como objetivo a identificação das linhas.
e) O Cassandra foi criado pelo próprio Google para atender suas necessidades, depois a empresa passou a comercializar o banco de dados para a comunidade.

2. Assinale a afirmativa incorreta em relação à comparação do modelo de dados orientado a colunas para o modelo de dados relacional:

a) Os bancos de dados relacionais são orientados ou organizados em linhas (registros de uma tabela), pois os dados em cada linha de uma tabela são armazenados juntos, por atributos ou campos de uma tabela.
b) No banco de dados colunar, diferentemente da forma usada na estrutura do banco de dados relacional, os dados são organizados e armazenados em colunas.
c) Um banco de dados no modelo relacional corresponde a um *keyspace* no orientado a colunas
d) Uma instância de bancos de dados no relacional corresponde a um *schema* no modelo colunar.
e) Uma coluna no relacional tem os mesmos tipos de dados e é igual para todas as linhas; já no colunar, as colunas podem ser diferentes pelas linhas.

3. **Atividade do objetivo 3**

Dado o código CQL para criar uma tabela denominada pedido com a seguinte estrutura, marque a opção incorreta sobre o que foi apresentado:

CREATE TABLE pedido (

nrpedidouuid primary key ,

livrovarchar ,

autor set<text>) ;

INSERT INTO pedido (nrpedido, livro, autor)

VALUES (uuid(), 'NOSQL Essencial', { 'Pramod J. Sadalage', 'MartinFowler' }) ;

a) É criada uma tabela no um identificar universal e o valor gerado pela função uuid().
b) Há um atributo do tipo set, mas não pode ser criado no Cassandra.
c) O insert apresenta o cadastro dos nomes de dois autores.
d) A tabela foi criada, mas também poderia ser definida como uma COLUMNFAMILY.
e) O uso do tipo de dado uuid em uma chave primária é indicado por ser único.

Notas
Referências

ELMASRI, R.; NAVATHE, S.B. **Sistemas de banco de dados**. 6.ed. São Paulo: Pearson Education do Brasil, 2018.

SADALAGE, P.J.; FOWLER, M. **NoSQL essencial** – um guia conciso para o mundo emergente da persistência poliglota. 1.ed. São Paulo: Novatec, 2013.

Próxima aula

- Comandos CQL para manipulação de dados: Insert, update e delete;
- Comandos CQL para consultas simples e com o uso de funções;
- Comandos CQL para consultas envolvendo agregações.

Explore mais

Para aprofundar seus conhecimentos, fica a sugestão de leitura do Livro Sistema de banco de dados de Elmasri, R.; Navathe, S.B.; Sistemas de Banco de Dados no do Cap. 24.5: Sistemas NOSQL baseados em documentos e MongoDB.

De modo e entender com mais detalhes do modelo orientado a documentos, faça a leitura do Capítulo 10 do livro Pramod J.

Sadalage, Martim Fowler. NoSQL Essencial - Um Guia Conciso para o Mundo Emergente da Persistência Poliglota. 1. São Paulo: Novatec, 2013.