

Apache Cassandra

SCC0241 – Laboratório de Base de Dados

Apresentado por:

Érica Ribeiro Filgueira dos Santos - 11836351

Thiago Prado Dalla Dea - 12691710

Sumário

01 Introdução

02 Finalidade

03 Paradigma

04 Operação

05 Segurança

06 Prós e Contras

07 Linguagem

08 Instalação

09 Configuração

10 Demonstração

Introdução

- O Apache Cassandra é um banco de dados NoSQL de código aberto escalável e robusto, projetado para lidar com grandes quantidades de dados distribuídos sem comprometer o desempenho ou a disponibilidade.
- Criado pelo Facebook, sua arquitetura foi inspirada pelo DynamoDB da Amazon e seu modelo de dados foi baseado no BigTable do Google.
- Arquitetura baseada em um modelo *peer-to-peer* descentralizado, onde cada nó no cluster tem igual importância.
- Adequado para aplicativos que exigem baixa latência, alto rendimento e escalabilidade contínua.

Finalidade

Devido aos seus muitos recursos robustos, o Cassandra pode ser usado em uma variedade de cenários diversos.

- **Sistemas de gerenciamento de conteúdo (CMS)**: escalabilidade e flexibilidade no gerenciamento de dados não estruturados ou semiestruturados
- **Sistemas de catálogo e inventário**: eficiência no manuseio de grandes conjuntos de dados, design de esquema dinâmico e flexível e acesso de baixa latência
- **Cargas de trabalho de escrita intensiva**: capacidade de lidar com muitas operações de gravação em nós distribuídos, mantendo o desempenho de baixa latência.

Finalidade

- **Sistemas de catálogo e inventário:** eficiência no manuseio de grandes conjuntos de dados, design de esquema dinâmico e flexível e acesso de baixa latência
- **Registro e rastreamento de eventos:** arquitetura tolerante a falhas e escalável garante captura e análise de dados confiáveis
- **Filas de mensagens e plataformas de comunicação:** capacidade de lidar com operações de gravação de alto rendimento e baixa latência

Finalidade

- Armazenamento de dados de séries temporais
- Análise em tempo real
- Mecanismos de recomendação
- Integração com Big Data

cassandra

Apache Cassandra

Paradigma

Arquitetura

Banco de Dados Relacional:

<u>sku_produto</u>	<u>nome_produto</u>	<u>preco_produto</u>
1	Tênis	100
2	Bola	50
3	Camisa	200
4	<u>Bike</u>	900

Banco de Dados Colunar:

<u>sku_produto</u>		<u>nome_produto</u>		<u>preco_produto</u>	
<u>id</u>	<u>value</u>	<u>id</u>	<u>value</u>	<u>id</u>	<u>value</u>
0	1	0	Tênis	0	100
1	2	1	Bola	1	50
2	3	2	Camisa	2	200
3	4	3	<u>Bike</u>	3	900

O Cassandra é um banco de dados não relacional e colunar.

Paradigma

Arquitetura

- Cassandra oferece algumas garantias sobre sua escalabilidade, disponibilidade e confiabilidade.
- Para entender as limitações de um sistema de armazenamento em um ambiente no qual um certo nível de falha de partição de rede é esperado e levado em consideração ao projetar o sistema, é importante apresentar o teorema CAP.

Paradigma

Arquitetura

- De acordo com o teorema CAP, não é possível que um armazenamento de dados distribuído forneça mais de duas das seguintes garantias:
 - **Consistency**: implica que toda leitura recebe a escrita ou erros mais recentes
 - **Availability**: implica que cada solicitação recebe uma resposta
 - **Partition tolerance**: refere à tolerância de um sistema de armazenamento à falha de uma partição de rede.
- O Cassandra é considerado um banco AP (Availability / Partition Tolerance) por priorizar a alta disponibilidade.

Paradigma

Arquitetura

O Cassandra oferece as seguintes garantias:

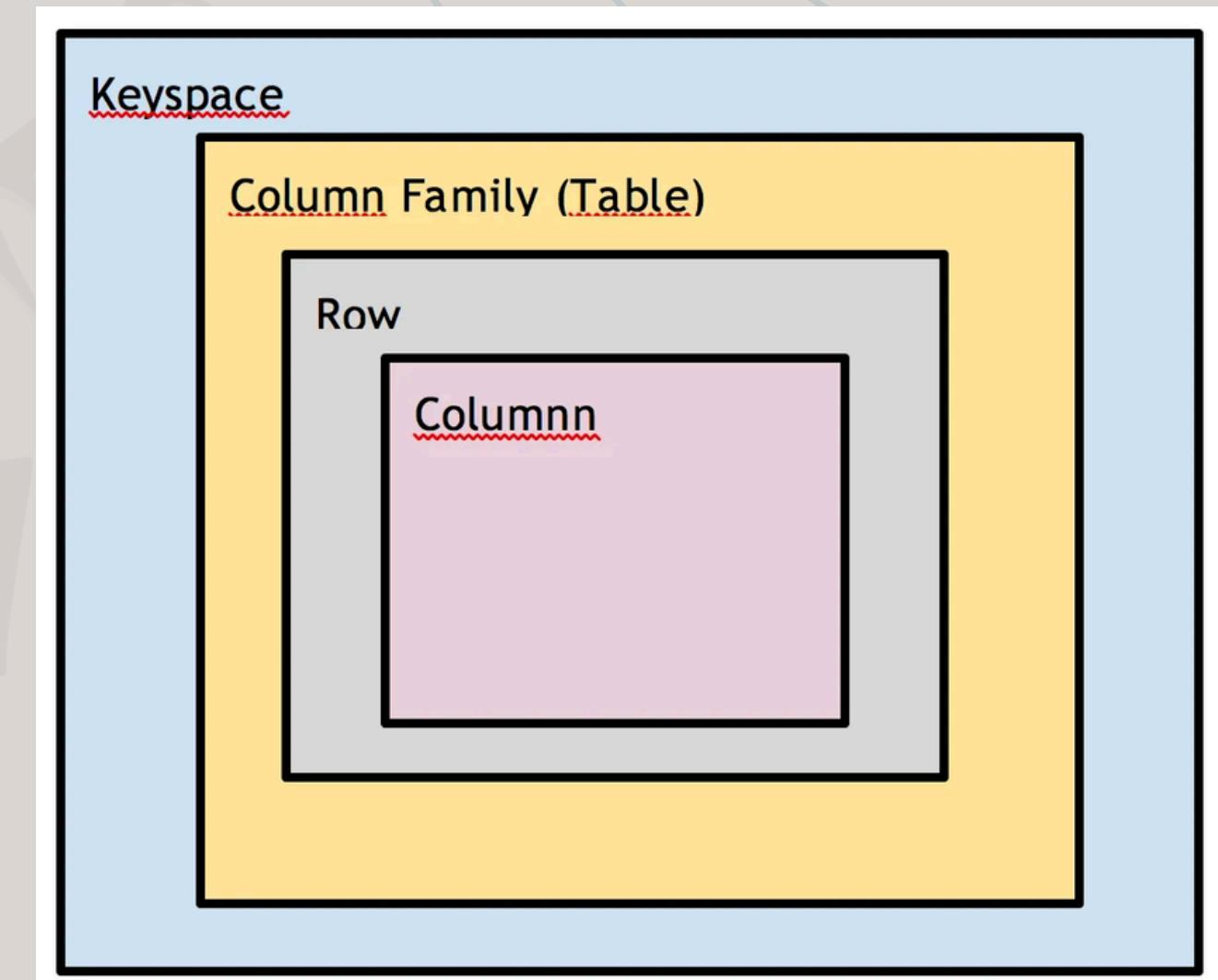
- Alta escalabilidade
- Alta disponibilidade
- Durabilidade
- Consistência eventual de escritas em uma única tabela
- Transações leves com consistência linearizável
- Gravações em lote em várias tabelas têm garantia de sucesso total ou nenhum sucesso
- Índices secundários têm garantia de consistência com os dados de suas réplicas locais

Paradigma

Arquitetura

O Cassandra é segmentado em:

- **Keyspace**: define como um conjunto de dados é replicado por datacenter
- **Table**: é composta por linhas e colunas; é particionadas baseado nas colunas fornecidas na *partition key*
- **Row**: contém uma coleção de colunas identificadas por uma chave primária
- **Column**: dado com tipo que pertence a uma linha



Paradigma

Modelagem de Dados

O modelo de dados que você usa é o fator mais importante para o seu sucesso com Cassandra. — Patrick McFadin

- **Modelagem orientada para consulta**
- Os padrões de acesso aos dados e as consultas da aplicação determinam a estrutura e a organização dos dados que são usados para projetar as tabelas do banco de dados.

Paradigma

Modelagem de Dados

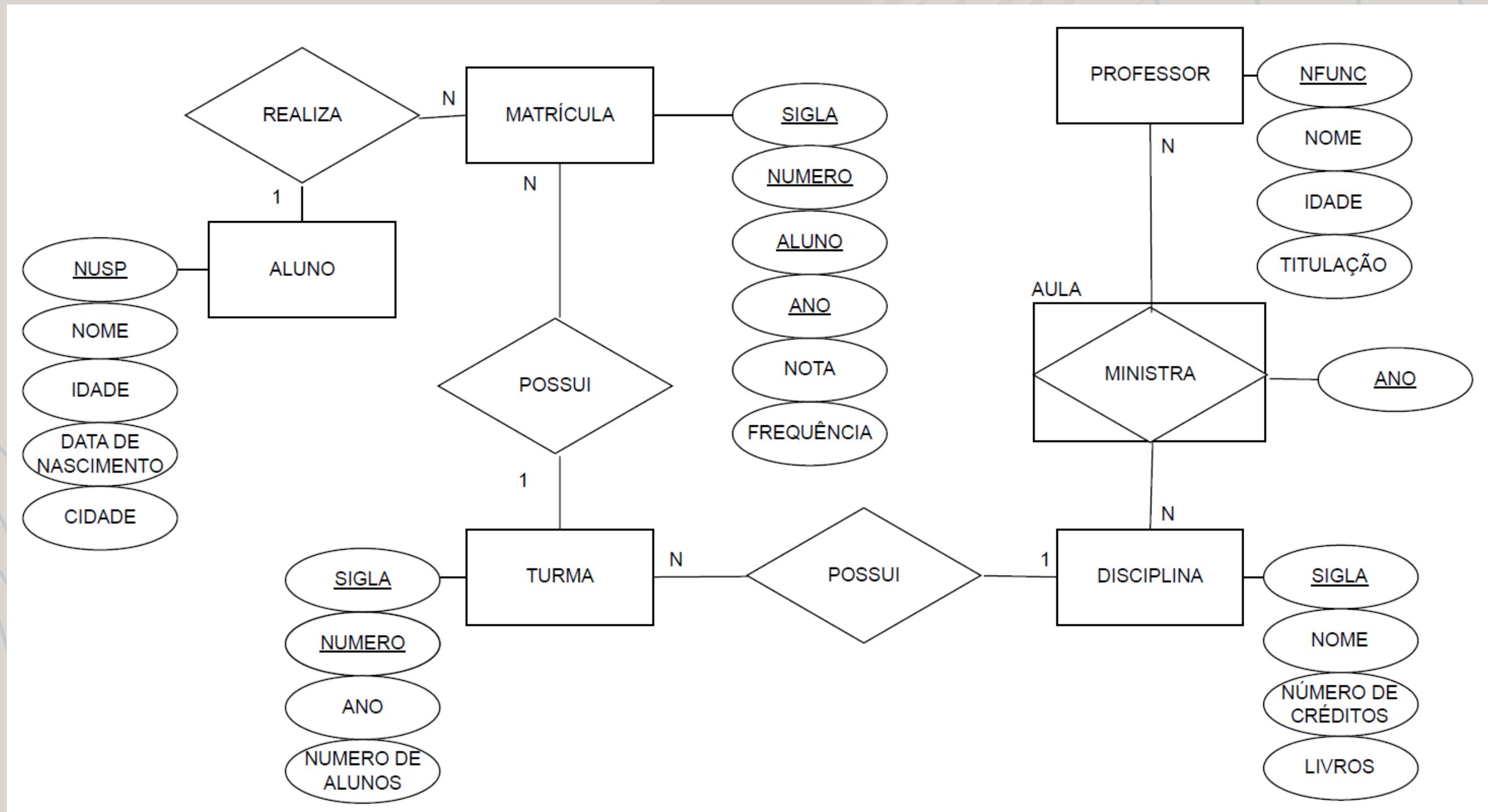
Diferenças de design entre RDBMS e Cassandra

- Sem junções
- Sem integridade referencial
- Desnormalização
- Design que prioriza a consulta
- Ordenação é uma decisão de design

Apache Cassandra

Paradigma

Modelagem de Dados Modelagem Conceitual



Paradigma

Modelagem de Dados
Definir consultas

Q1: Obter todas as disciplinas que um aluno está cursando/cursou, com nota e frequência.
(historico do aluno)

Q2: Obter detalhes de uma disciplina específica.

Q3: Obter todos os alunos matriculados em uma disciplina específica, com suas notas e frequência.

Q4: Obter detalhes de um aluno específico.

Q5: Obter todas as disciplinas ministradas por um professor em um determinado ano.

Q6: Obter todas as turmas de uma disciplina específica em um ano.

Q7: Obter detalhes de um professor.

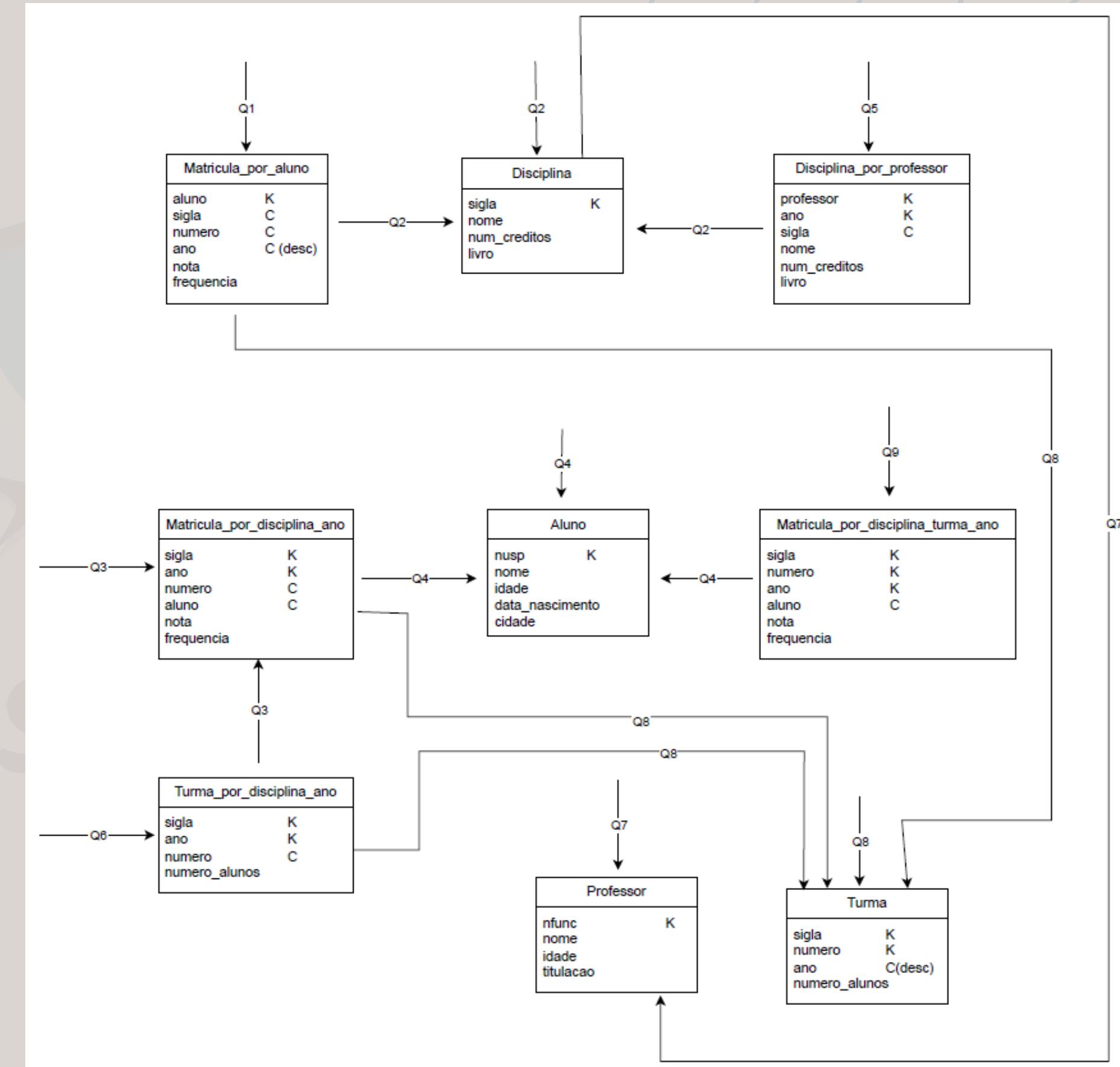
Q8: Obter detalhes de uma turma.

Q9: Obter todos os alunos de uma turma especifica.

Apache Cassandra

Paradigma

Modelagem de Dados Modelagem Lógica



Apache Cassandra

Paradigma

Modelagem de Dados Modelagem Física

Universidade Keyspace

Matricula	
aluno	K
sigla	C
numero	C
ano	C (desc)
nota	
frequencia	

Aluno	
nusp	K
nome	
idade	
data_nascimento	
cidade	

Disciplina	
sigla	K
nome	
num_creditos	
livro	

Turma	
sigla	K
numero	K
ano	C(desc)
numero_alunos	

Professor	
nfunc	K
nome	
idade	
titulacao	

Matricula_por_disciplina_ano	
sigla	K
ano	K
numero	C
aluno	C
nota	
frequencia	

Disciplina_por_professor	
professor	K
ano	K
sigla	C
nome	
num_creditos	
livro	

Matricula_por_disciplina_turma_ano	
sigla	K
numero	K
ano	K
aluno	C
nota	
frequencia	

Turma_por_disciplina_ano	
sigla	K
ano	K
numero	C
numero_alunos	

Operação

Backup e Recuperação

- Apache Cassandra armazena dados em SSTables (arquivos imutáveis do disco)
 - Dados inicialmente em memória -> Transferência para o disco em SSTable
- Principais propósitos:
 - Armazenar uma cópia de dados para maior durabilidade
 - Restauração de tabelas em caso de uma falha de nó/partição/rede
 - Transferir os arquivos SSTable para uma máquina diferente, para portabilidade

Operação

Backup e Recuperação: Snapshots

- Snapshots são cópias dos arquivos SSTable de uma tabela em um dado momento, criadas com hard links. O DDL da tabela também é armazenado.
- Pode ser feita pelo usuário ou automaticamente:
 - Configuração *snapshot_before_compaction* no arquivo *cassandra.yaml* controla criação automática antes de cada compaction (por padrão, *false*).
 - *auto_snapshot*: Configuração para criar snapshots automáticos antes de truncagem de keyspaces ou exclusão de tabelas (ativado por padrão).

Operação

Backup e Recuperação: Incremental Backup

- São cópias dos arquivos SSTable de uma tabela, criadas por hard links durante o flush de *memtables* para SSTables.
- Geralmente combinados com snapshots para reduzir o tempo e o espaço de armazenamento de backups.
- Quando ativado, cria hard links para cada SSTable no subdiretório backups/ dentro dos dados do keyspace.

Operação

Integração com Linguagens de Programação

- O Cassandra oferece drivers oficiais para diversas linguagens.
 - Java
 - Python
 - Node.js
- Drivers suportam operações de leitura e escrita, autenticação, consistência e balanceamento de carga.
- Possibilidade de uso em diversas arquiteturas de aplicação.

Operação

Análise de Dados em Larga Escala e Streaming

- Ingestão de dados em tempo real
 - Apache Kafka.
 - Captura de dados em tempo real.
 - Apache Spark e Apache Flink
 - Análise avançada e enriquecimento de dados diretamente do banco de dados.
 - Criação de pipelines de dados sofisticados.

Operação

Arquitetura de Nuvem e DevOps

- Amplamente suportado nas principais plataformas de nuvem
 - AWS, GCP e Azure.
- Apache Cassandra as a Service
 - Oferecimento do Cassandra como serviço (DBaaS)
 - DataStax Astra: retira a necessidade de gerenciar clusters manualmente, simplifica o gerenciamento e a escalabilidade, permitindo foco total no desenvolvimento e análise de dados.

Segurança

Principais funcionalidades:

- Autenticação e Autorização
 - Controle de Acesso Baseado em Papéis
- Auditoria de Acesso
- Conexões seguras (TLS/SSL)
- Integração com MBeans

Segurança

Autenticação e Autorização

- O controle de acesso e autenticação é gerenciado através de papéis.
- Os papéis no Cassandra permitem definir permissões hierárquicas como SELECT, MODIFY e CREATE, que controlam quem pode visualizar ou modificar dados.
- Cada papel pode representar um usuário ou um grupo, facilitando o controle de acesso.
- Garante que apenas usuários autorizados acessem informações confidenciais.
 - Permissões incluem operações como SELECT, MODIFY, e CREATE

Segurança

Auditoria de Acesso

- Garante conformidade e segurança operacional.
- Fornece uma trilha de auditoria completa.
- Permite registrar atividades e alterações no banco de dados.
- As informações de auditoria são configuráveis e são armazenadas em arquivos de log para serem revisadas por administradores.

Segurança

Conexões Seguras (TLS/SSL)

- Conexões com TLS/SSL são usadas para criptografar a comunicação entre os nós do cluster e entre clientes e servidores.
- Os administradores devem gerar e configurar certificados TLS (Trusted Certificate) e de chave pública.
- Para criptografia entre nós do cluster, configura-se a criptografia no arquivo `cassandra.yaml`.
 - Validação da identidade dos nós antes de permitir a comunicação.
- A criptografia de cliente para servidor é configurada com um certificado SSL para autenticação de clientes ao cluster.
 - Apenas clientes autenticados acessam o banco de dados

Segurança

Integração com MBeans

- Possibilidade de configurar permissões em MBeans (Managed Bean).
 - Configurações do controle de acesso e operações
- Restringe operações administrativas a usuários autorizados (admins).
 - Proteção de tarefas críticas
- Métricas e gerenciamento centralizados para segurança em tempo real

Prós e Contras

Prós

- Escalabilidade
- Alta Disponibilidade
- Não tem ponto único de falha
- Modelo de dados flexível
- Alta performance

Prós e Contras

Contras

- Complexidade
- Linguagem
- Junções limitadas
- Modelo de dados flexível
- Alta sobrecarga de armazenamento
- Consistência eventual

Linguagem Declarativa

Cassandra Query Language (CQL)

CQL oferece suporte a vários recursos avançados em um conjunto de dados particionado, como:

- Tipos de coleção, incluindo conjuntos, mapas e listas
- Tipos, tuplas, funções e agregações definidas pelo usuário
- Indexação anexada ao armazenamento (SAI) para índices secundários
- Índices secundários locais
- *Materialized views*

Linguagem Declarativa

Cassandra Query Language (CQL)

Tipos de dados

CQL é uma linguagem tipada e oferece suporte a um rico conjunto de tipos de dados, incluindo tipos nativos, tipos de coleção, tipos definidos pelo usuário, tipos de tupla e tipos personalizados.

Linguagem Declarativa

Cassandra Query Language (CQL)

Tipos de dados nativos

ascii
bigint
blob
boolean
counter

date
decimal
double
duration
float
inet

int
smallint
text
time
timestamp
timeuuid

tinyint
uuid
varchar
varint
vector

Linguagem Declarativa

Cassandra Query Language (CQL)

Data Definition (DDL)

CQL armazena dados em tabelas, cujo esquema define o layout dos dados na tabela. As tabelas estão localizadas em *keyspaces*. Um keyspace define opções que se aplicam a todas as tabelas do keyspace. A estratégia de replicação é uma opção importante de keyspace, assim como o fator de replicação

Linguagem Declarativa

Cassandra Query Language (CQL)

Data Definition (DDL)

CREATE KEYSPACE

CREATE KEYSPACE [IF NOT EXISTS] `keyspace_name` WITH options

CREATE KEYSPACE `universidade`

WITH replication = {'class': 'SimpleStrategy', 'replication_factor' : 3};

Linguagem Declarativa

Cassandra Query Language (CQL)

Data Definition (DDL)

USE

USE keyspace_name

ALTER KEYSPACE

ALTER KEYSPACE [IF EXISTS] keyspace_name WITH options

DROP KEYSPACE

DROP KEYSPACE [IF EXISTS] keyspace_name

Linguagem Declarativa

Cassandra Query Language (CQL)

Data Definition (DDL)

CREATE TABLE

```
CREATE TABLE [ IF NOT EXISTS ] table_name '('  
    column_definition ( ',' column_definition )*  
    [ ',' PRIMARY KEY '(' primary_key ')' ]  
'' [ WITH table_options ]
```

Linguagem Declarativa

Cassandra Query Language (CQL)

Data Definition (DDL)

CREATE TABLE

```
CREATE TABLE matricula (
    sigla TEXT,
    numero INT,
    aluno BIGINT,
    ano INT,
    nota FLOAT,
    frequencia FLOAT,
    PRIMARY KEY ((aluno), ano, sigla, numero)
) WITH CLUSTERING ORDER BY (ano DESC);
```

Linguagem Declarativa

Cassandra Query Language (CQL)

Data Definition (DDL)

CREATE TABLE

Primary key

- partition key

Uma partição é o conjunto de linhas que compartilham o mesmo valor para sua chave de partição, e define a localização dos dados dentro de um cluster Cassandra.

- clustering columns

As colunas de agrupamento adicionam exclusividade a uma linha em uma tabela, além de definirem a ordem de *cluster* para a partição dessa tabela.

Linguagem Declarativa

Cassandra Query Language (CQL)

Data Definition (DDL)

CREATE TABLE

Table options

- clustering order by

Influencia como as consultas podem ser feitas na tabela.

Linguagem Declarativa

Cassandra Query Language (CQL)

Data Manipulation (DML)

SELECT

```
SELECT [JSON | DISTINCT ]( select_clause | '*' )
FROM `table_name`
[ WHERE `where_clause` ]
[ GROUP BY `group_by_clause` ]
[ ORDER BY `ordering_clause` ]
[ PER PARTITION LIMIT (`integer` | `bind_marker` ) ]
[ LIMIT (`integer` | `bind_marker` ) ]
[ ALLOW FILTERING ]
```

Apache Cassandra

Linguagem Declarativa

Cassandra Query Language (CQL)

Data Manipulation (DML)

SELECT

Q1: SELECT sigla, numero, nota, frequencia FROM matricula_por_aluno WHERE aluno = 101;

Q2: SELECT nome, ncred, professor, livro FROM disciplina WHERE sigla = 'MAT101';

Q3: SELECT * FROM matricula_por_disciplina_ano WHERE sigla = 'MAT101' and ano = 2022;

Q4: SELECT nome, idade, data_nascimento, cidade FROM aluno WHERE nusp = 101;

Q5: SELECT sigla FROM disciplinas_por_professor WHERE nfunc = 1001 AND ano = 2022;

Q6: SELECT * FROM turmas_por_disciplina_ano WHERE sigla = 'MAT101' AND ano = 2022;

Q7: SELECT nome, idade, titulacao FROM professor WHERE nfunc = 1001;

Q8: SELECT ano, numalunos FROM turma WHERE sigla = 'MAT101' AND numero = 1;

Q9: SELECT aluno FROM matricula_por_disciplina_turma_ano WHERE sigla = 'MAT101' AND numero = 1 AND ano = 2022;

Linguagem Declarativa

Cassandra Query Language (CQL)

Data Manipulation (DML)

UPDATE

```
UPDATE table_name  
[ USING update_parameter ( AND update_parameter )* ]  
SET assignment( ',' assignment )*  
WHERE where_clause  
[ IF ( EXISTS | condition ( AND condition)* ) ]
```

Linguagem Declarativa

Cassandra Query Language (CQL)

Data Manipulation (DML)

DELETE

```
DELETE [ simple_selection ( ',' simple_selection ) ]
FROM table_name
[ USING update_parameter ( AND update_parameter# )* ]
WHERE where_clause
[ IF ( EXISTS | condition ( AND condition)* ) ]
```

Linguagem Declarativa

Cassandra Query Language (CQL)

Data Manipulation (DML)

BATCH

```
BEGIN [ UNLOGGED | COUNTER ] BATCH  
[ USING update_parameter( AND update_parameter)* ]  
modification_statement ( ';' modification_statement )*  
APPLY BATCH
```

Linguagem Declarativa

Cassandra Query Language (CQL)

Materialized views

CREATE MATERIALIZED VIEW

```
CREATE MATERIALIZED VIEW [ IF NOT EXISTS ] view_name  
AS select_statement  
PRIMARY KEY '(' primary_key ')'  
WITH table_options
```

Linguagem Declarativa

Cassandra Query Language (CQL)

Materialized views

ALTER MATERIALIZED VIEW

ALTER MATERIALIZED VIEW [IF EXISTS] view_name WITH table_options

DROP MATERIALIZED VIEW

DROP MATERIALIZED VIEW [IF EXISTS] view_name

Instalação

Pré-requisitos

- Versão atualizada do Java 8, Java 11 ou Java 17
- Para utilizar o shell CQL, cqlsh, Python 3.8-3.11

Métodos de Instalação

- Imagem do Docker
- Arquivo binário Tarball
- Instalação de pacotes (RPM, YUM)

Instalação

Instalar como pacote Debian

- Adicionar o repositório do Apache Cassandra a /etc/apt/sources.list.d/cassandra.sources.list

```
$ echo "deb [signed-by=/etc/apt/keyrings/apache-cassandra.asc]
https://debian.cassandra.apache.org 50x main" | sudo tee -a
/etc/apt/sources.list.d/cassandra.sources.list deb https://debian.cassandra.apache.org 50x main
```

- Adicionar as chaves do repositório:

```
$ curl -o /etc/apt/keyrings/apache-cassandra.asc
https://downloads.apache.org/cassandra/KEYS
```

Instalação

Instalar como pacote Debian

- Atualizar a lista de pacotes:
`$ sudo apt-get update`
- Instalar Cassandra com APT:
`$ sudo apt-get install cassandra`
- Confira a situação de Cassandra:
`$ nodetool status`
- Verificar status do Cassandra:
`$ sudo service cassandra status`
- Inicializar o Cassandra:
`$ sudo service cassandra start`
- Utilizar o CQL shell:
`$ cqlsh`

Configuração

- **cassandra.yaml**: arquivo de configuração principal do Cassandra
- **cassandra-env.sh**: definir variáveis de ambiente
- **cassandra-rackdc.properties** OU **cassandra-topology.properties**: definir informações de rack e datacenter para um cluster
- **logback.xml**: configuração de registro, incluindo níveis de registro
- **jvm-***: vários arquivos de configuração JVM para servidor e clientes
- **commitlog_archiving.properties**: define parâmetros de arquivamento para o commitlog
- **cqlshrc.sample**: como o shell CQL, cqlsh, pode ser configurado

Demonstração Prática

Conexão com o Banco de Dados

```
CqlSession session = CqlSession.builder()  
    .addContactPoint(new InetSocketAddress("127.0.0.1", 9042))  
    .withLocalDatacenter("datacenter1")  
    .build();
```

Executar queries

```
ResultSet res = session.execute(<query>);
```

Apache Cassandra

Obrigado pela atenção!

cassandra