



**MBA<sup>+</sup>**

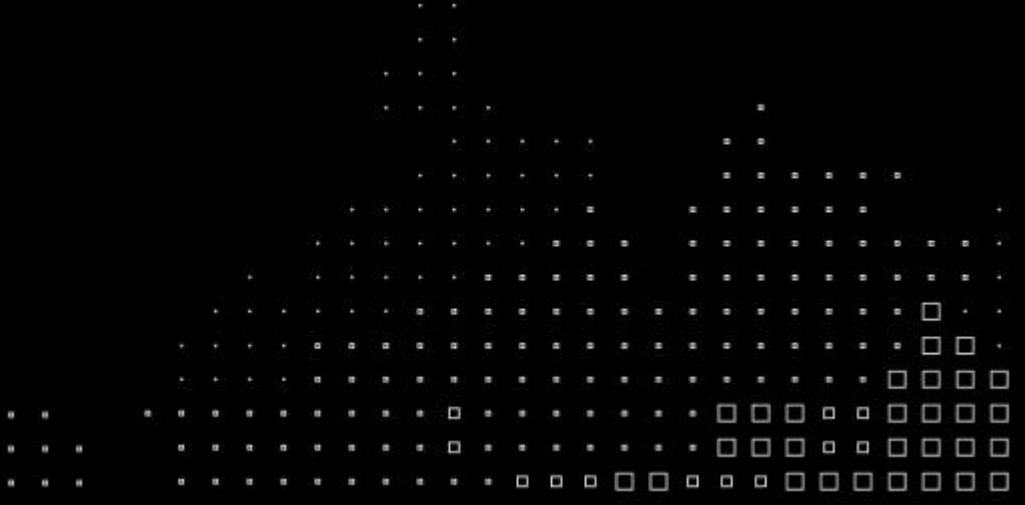
**Data Science &  
Artificial Intelligence**



**MBA<sup>+</sup>****Data Architecture,  
Integration and Ingestion**

Prof.: Ivan Gancev

Email: [profivan.gancev@fiap.com.br](mailto:profivan.gancev@fiap.com.br)



# Data Architecture, Integration and Ingestion

(O que vamos explorar?)

## Aula 1 – 12/abr (qua)

- Pilares de arquitetura: persistência, integração e consumo
- Estratégias de arquitetura
- Tipos de tratamentos e arquiteturas

## Aula 2 – 19/abr (qua)

- Exemplos de Bancos, diferenças e usos:
  - Bancos Relacionais
  - Bancos Colunares

## Aula 3 – 26/abr (qua)

- Exemplos de Bancos, diferenças e usos:
  - Bancos de documentos
  - Bancos chave-valor
  - Bancos de Grafos

## Aula 4 – 03/mai (qua)

- Ingestão de dados, tratamentos e manipulações
- Pipeline de dados, governança e qualidade
- Integração de dados
  - Cargas batch, ETL, vantagens e desvantagens

## Aula 5 – 10/mai (qua)

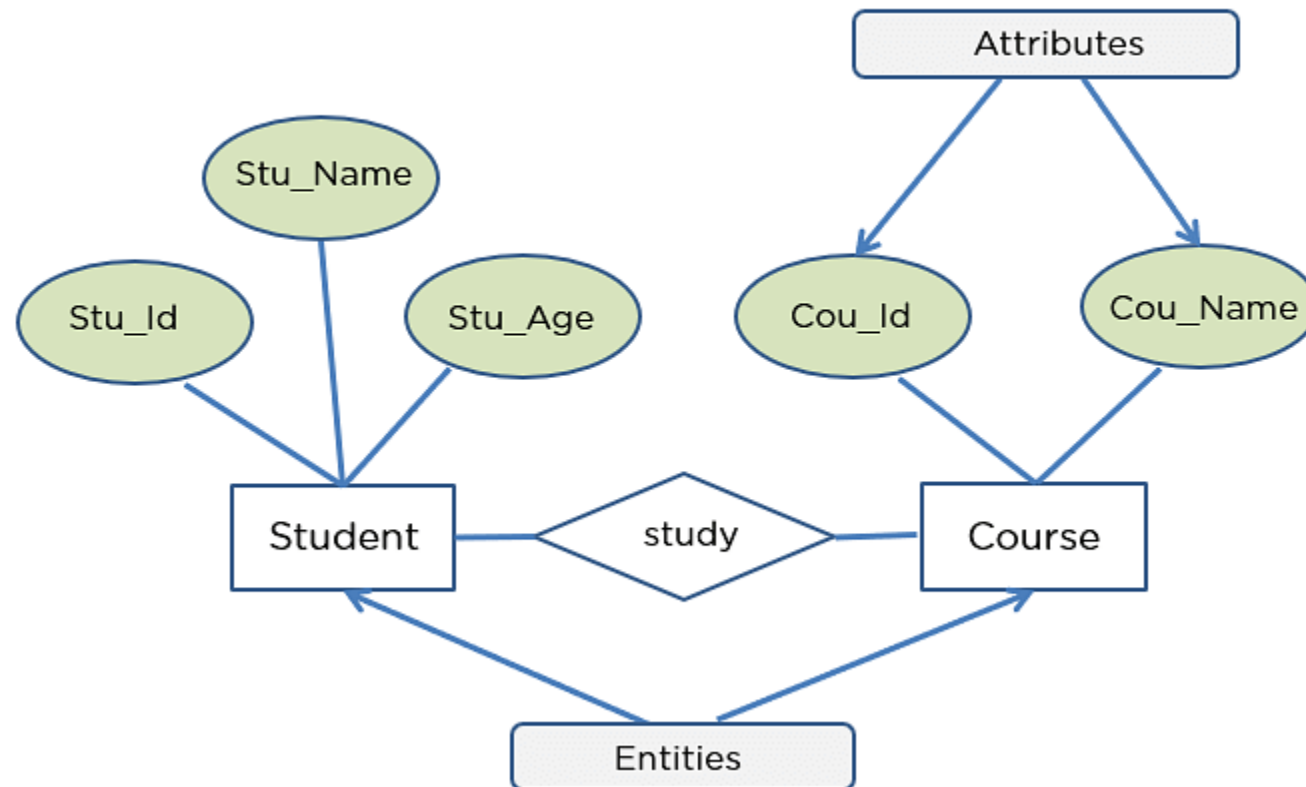
- Eventos, APIs, NRT e casos de uso
- Arquiteturas para analytics
- Boas práticas, recomendações e cuidados

# Bancos de dados relacionais (SQL)

# Características de bancos relacionais

- **Chaves primárias** nas tabelas que garantem a unicidade de um registro em uma tabela
- **Chaves estrangeiras** que relaciona uma tabela com a chave primária de outra
- Propriedades **ACID**:
  - **Atomicidade** (Atomicity): Define que uma operação é executada com sucesso ou uma falha será apontada
  - **Consistência** (Consistency): Regras que garantem o estado de um dado
  - **Isolação** (Isolation): Uma operação em andamento não é visível por outros usuários até que seja efetivada
  - **Durabilidade** (Durability): Garante mudanças permanentes depois que uma transação é concluída
- Modelo de dados estruturado e pré-definido para armazenamento dos dados
- Linguagens padronizadas para definição, manipulação, controle e transação de dados

# Modelo relacional



45697056

# Structured Query Language (SQL)

FIAP

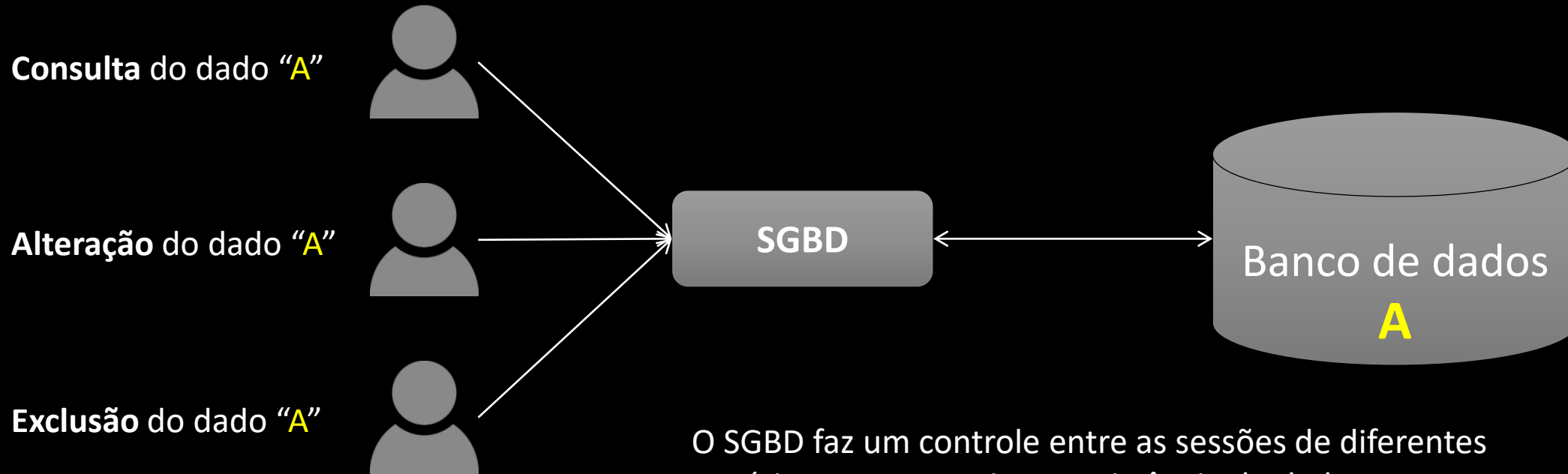
**DDL:** Data Definition Language → CREATE, ALTER, TRUNCATE, DROP, DESCRIBE, RENAME

**DML:** Data Manipulation Language → INSERT, UPDATE, DELETE, SELECT

**DCL:** Data Control Language → GRANT, REVOKE

**TCL:** Transaction Control Language → ROLLBACK, COMMIT

# Controle de sessões



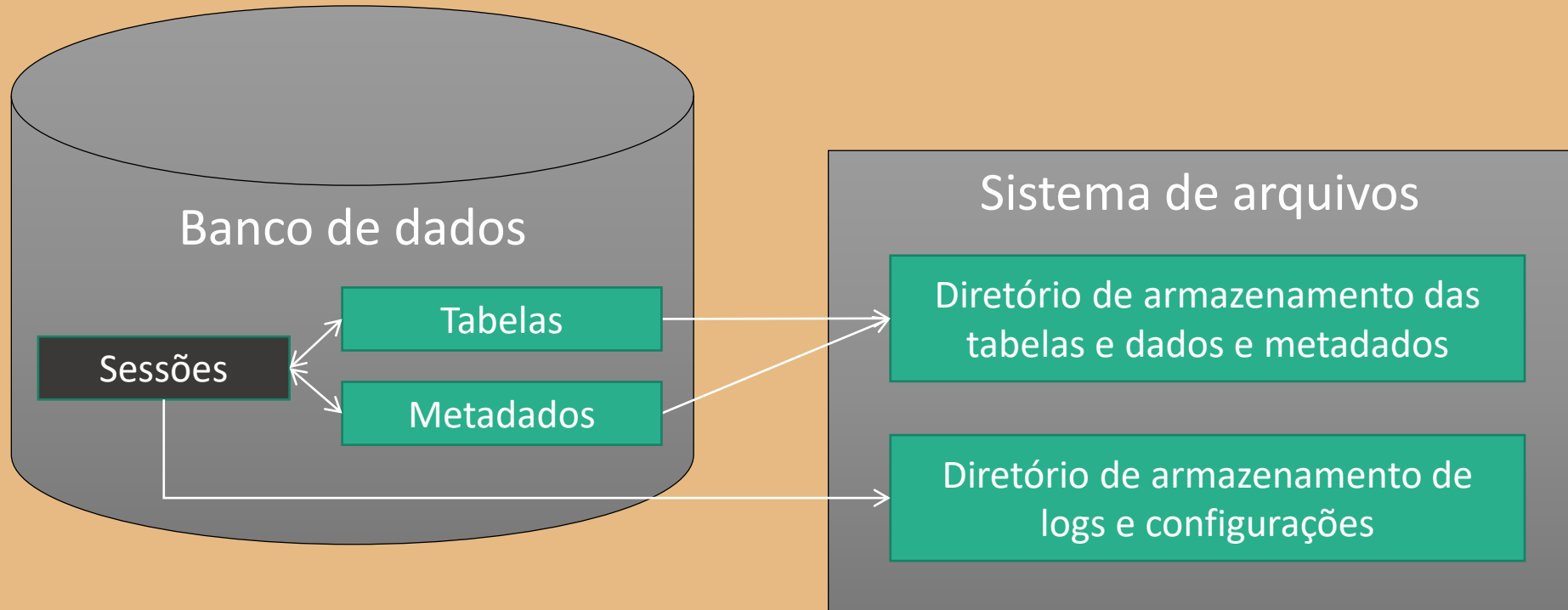
O SGBD faz um controle entre as sessões de diferentes usuários para garantir a consistência dos dados.

Há opções para consulta de dados com transações em andamento.



# Armazenamento

## Servidor



# Metadado



+

+

Metadado de um banco de dados é o “dado do dado”

Cada sistema de banco de dados possui tabelas de sistema que controlam a especificação de cada tabela, índices, configurações, permissões, usuários, senhas, parâmetros, sessões e demais informações que o SGBD usa para otimizar sua performance.

Estas tabelas são visíveis e acessíveis somente para usuários administradores

---

Ex.: O SGBD gerencia uma tabela interna que armazena os nomes de todas as tabelas existentes no banco

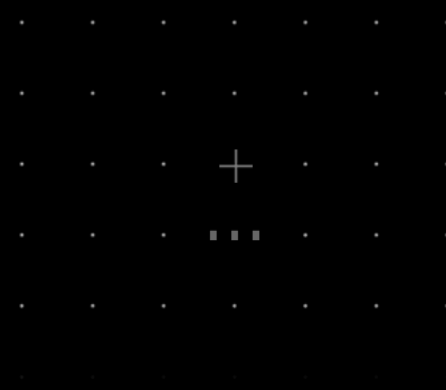
45697056

# Procedimentos

Alguns SGBDs implementam procedimentos com regras de negócio para tratamento dos dados.

São úteis para criação de processos batch, integrações, importações, etc

# Triggers



São rotinas disparadas a partir de um evento de manipulação de dados.

Disparadas de acordo com a configuração definida, como inclusão, alteração ou exclusão de um dado.

Ex: Quando um registro é inserido em uma tabela, uma trigger pode gravar um campo no registro incluído com o nome do usuário e a data da criação do registro



45697056

# Transações



Transações são conjuntos de operações de manipulação de dados que são efetivas ou desfeitas em conjunto.

Caso a transação seja concluída até o fim com sucesso, todas as operações são efetivas.

Caso contrário, as operações não são salvas e os dados voltam ao estado anterior ao início da transação

É útil quando há dependência funcional entre uma ou mais operações realizadas no banco.

Ex.: Suponha que para confirmar um pagamento seja necessário alterar 3 tabelas. Uma transação garante que as 3 tabelas serão alteradas para confirmar o pagamento.

# Quando usar um banco relacional

- ✓ Estrutura pré-definida de dados
- ✓ Consistência de dados mais importante do que escalabilidade e elasticidade
- ✓ Sem necessidade de consultas em tempo real
- ✓ Simplicidade e velocidade no desenvolvimento
- ✓ Acesso ao dado pelo usuário final facilitado
- ✓ Volumes de dados na casa de gigabytes ou terabytes

# Exemplos de bancos relacionais

FIAP

+



+



45697056

# Exercício:

Instalando e explorando um banco MySQL

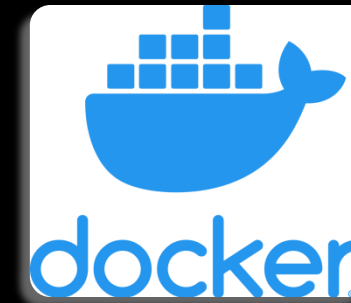


Ref.: <https://www.mysql.com/>



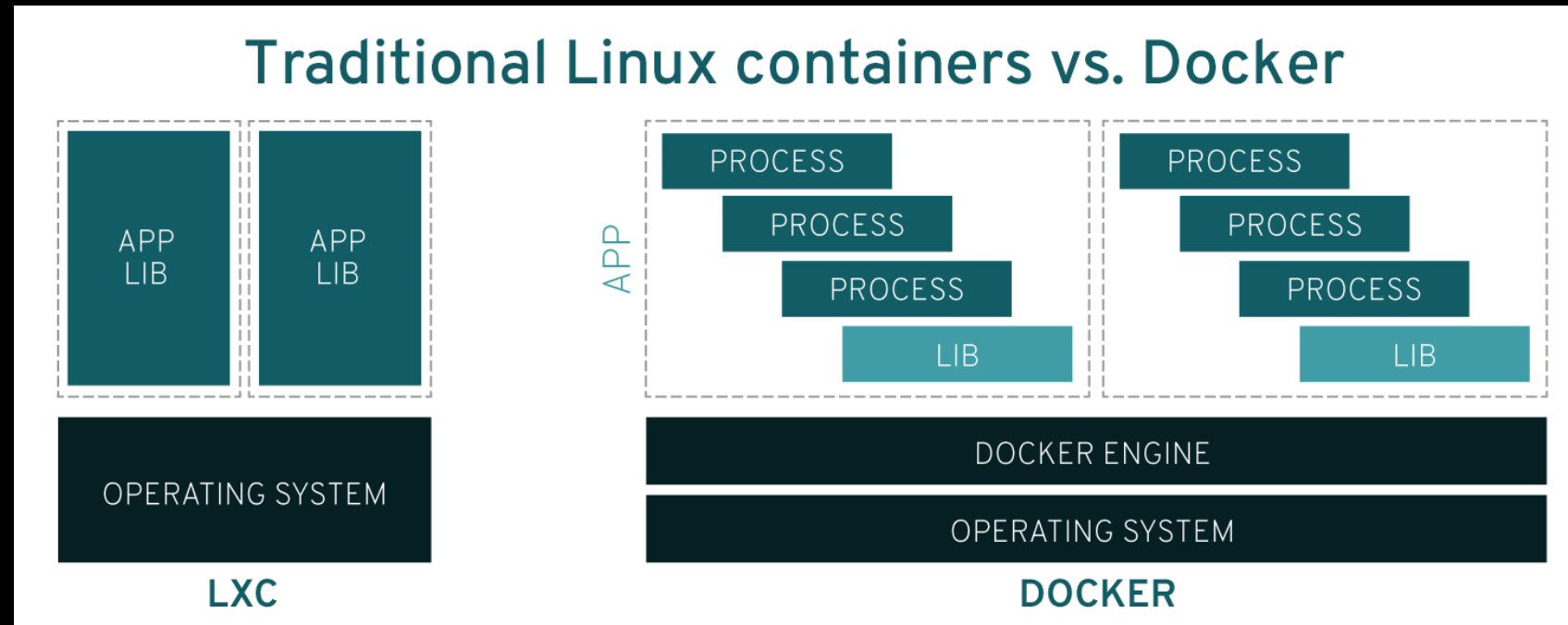
# Vamos usar Docker...

Para este exercício vamos usar um container.



Docker é um sistema de código aberto capaz de compartilhar recursos de um mesmo sistema operacional, contudo manter um isolamento de dependências de execução das aplicações

Uma **Imagem** é um conjunto de ferramentas, bibliotecas e aplicativos que uma vez instanciados em memória passam a ter o nome **Container**.



# Exercício MySQL

O Docker já está instalado e uma imagem chamada *ivangancev/ubuntusql:latest* já está disponível. Siga os passos iniciais abaixo:

1. Criar um Container a partir da imagem
2. Fazer a instalação do MySQL
3. Verificar o estado do serviço e iniciar o serviço MySQL
4. Acessar o CLI do Mysql
5. Criar um database chamado *db\_dts*
6. Acessar o database criado
7. Criar uma tabela chamada *aula\_mysql*
8. Inserir um registro na tabela criada
9. Consultar os registros da tabela *aula\_mysql*

Script disponibilizado  
com os comandos

45697056

# Continuação exercício MySQL...

## Usuários e acessos

10. Criar um usuário chamado *fiap* com a senha *f1@p*
11. Sair do CLI do MySQL
12. Acessar o CLI MySQL com o usuário *fiap* (\*)
13. Inserir um registro na tabela *aula\_mysql* (\*).
14. Sair do CLI e reconectar com o usuário anterior
15. Dar permissão ao usuário *fiap* na tabela *aula\_mysql*
16. Sair do CLI e reconectar com o usuário *fiap*
17. Inserir 1 registro na tabela
18. Consultar os registros da tabela *aula\_mysql*

Com qual usuário estava conectado antes?

O que houve neste passo e porquê?

45697056

# Continuação exercício MySQL...

## Manipulação de dados

- 19. Alterar um registro na tabela *aula\_mysql*
- 20. Consultar os registros da tabela *aula\_mysql*
- 21. Excluir 1 registro da tabela *aula\_mysql*
- 22. Consultar os registros da tabela *aula\_mysql* (\*)

Porque foi possível alterar e excluir registros?

## Armazenamento dos dados

- 23. Sair do CLI do MySQL
- ~~24. Navegar até o diretório do database *db\_dts*~~
- 25. Abrir o arquivo de dados da tabela *aula\_mysql* (\*)

É possível interpretar os dados do arquivo?

# Continuação exercício MySQL...

## Remoção dos acessos e dados

- 26. Abrir o CLI do MySQL com o usuário root
- 27. Revogue as permissões do usuário *fiap*
- 28. Remova a tabela *aula\_mysql*
- 29. Remova o database *db\_dts*

## Armazenamento dos dados

- 30. Sair do CLI do MySQL
- ~~31. Navegar até o diretório do database *db\_dts*<sup>(\*)</sup>~~

O que houve?

# Desafios

1. Crie um database e uma tabela de clientes (id, nome, idade, e-mail)
2. Insira alguns registros nessa tabela
3. Explore o comando select mudando filtros e campos selecionados
4. Explore o comando update, atualizando alguns registros
5. Atualize mais de um campo por execução, múltiplos registros
6. Use o comando truncate table
7. Pare o serviço do mysql e inicie novamente

# Bancos de dados não apenas relacionais (Not Only SQL / NoSQL)

# Surgimento de bancos NoSQL

**Com o crescimento nos volumes de dados gerados, tipos diferentes e escalabilidade global, a fragilidade dos bancos relacionais sobre esses aspectos favoreceram o surgimento de tecnologias capazes de entregar estas características e que ajudam a constituir o ecossistema de big data. Essas tecnologias ficaram conhecidas como Not Only SQL (NoSQL).**



# O Teorema de CAP

**C – Consistency / Consistência**

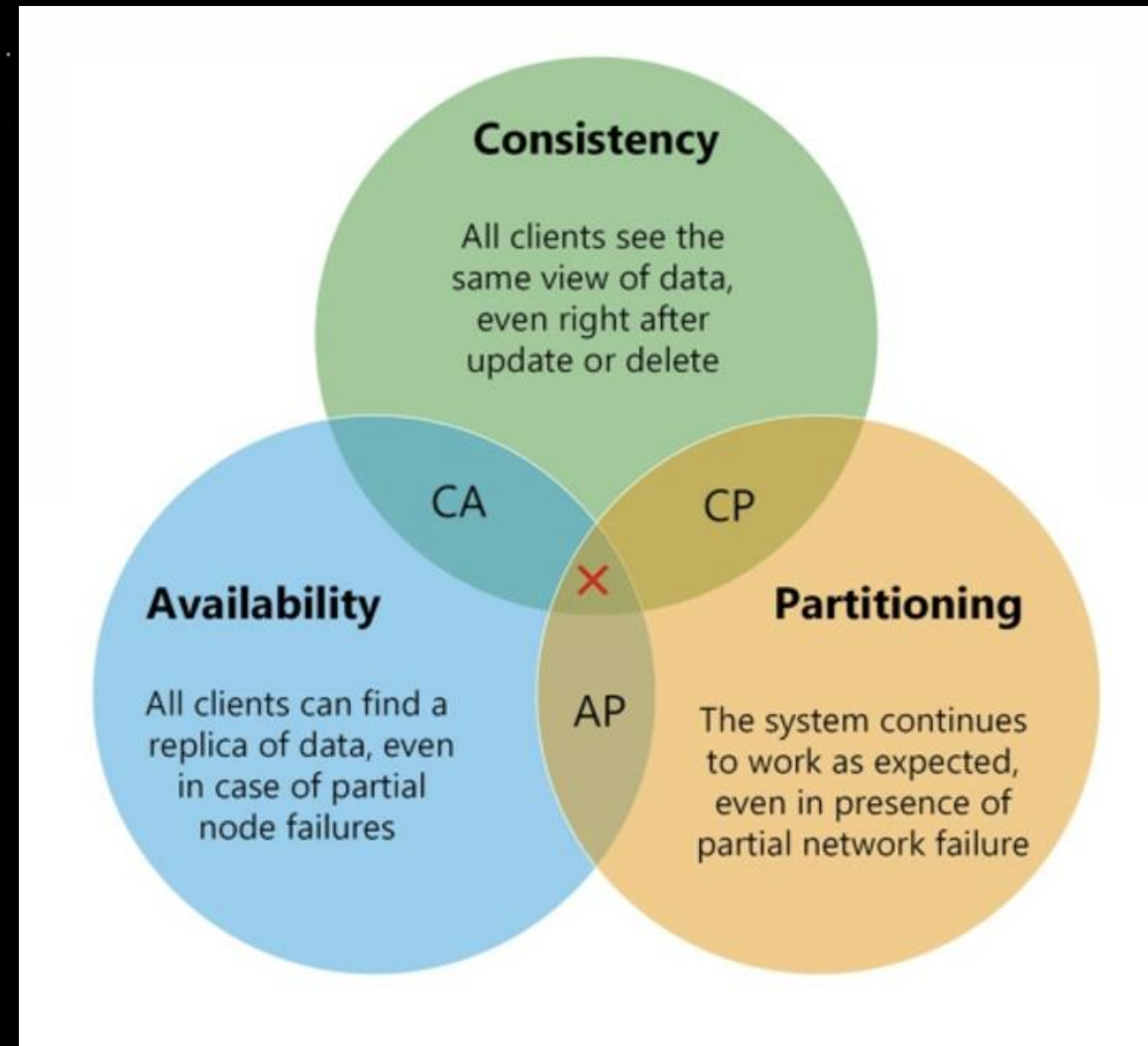
**A – Availability / Disponibilidade**

**P – Partitioning / Particionamento**

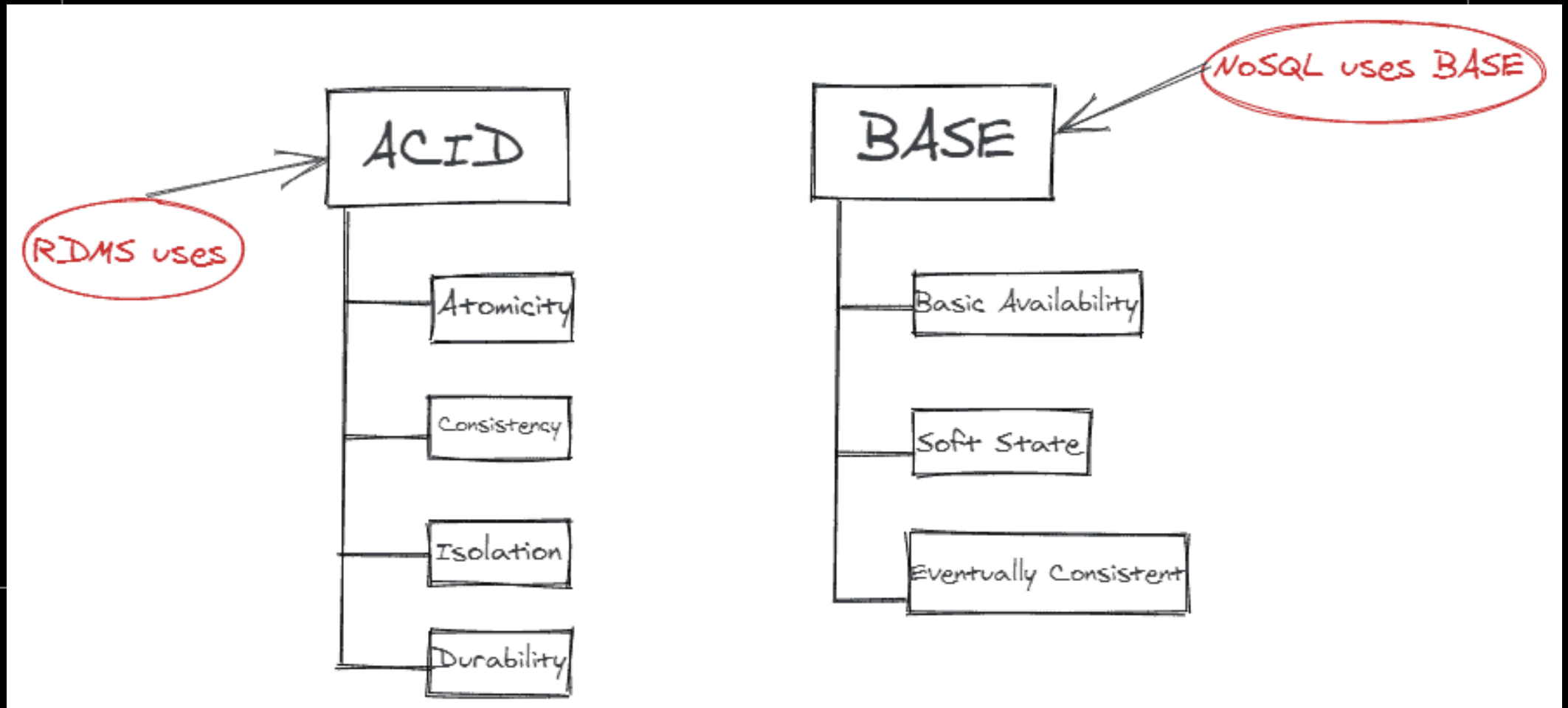
O Teorema de CAP afirma que um sistema distribuído de dados não é capaz de entregar simultaneamente mais de duas dessas capacidades.

Foi criado pela primeira vez pelo professor Eric A. Brewer em 2000.

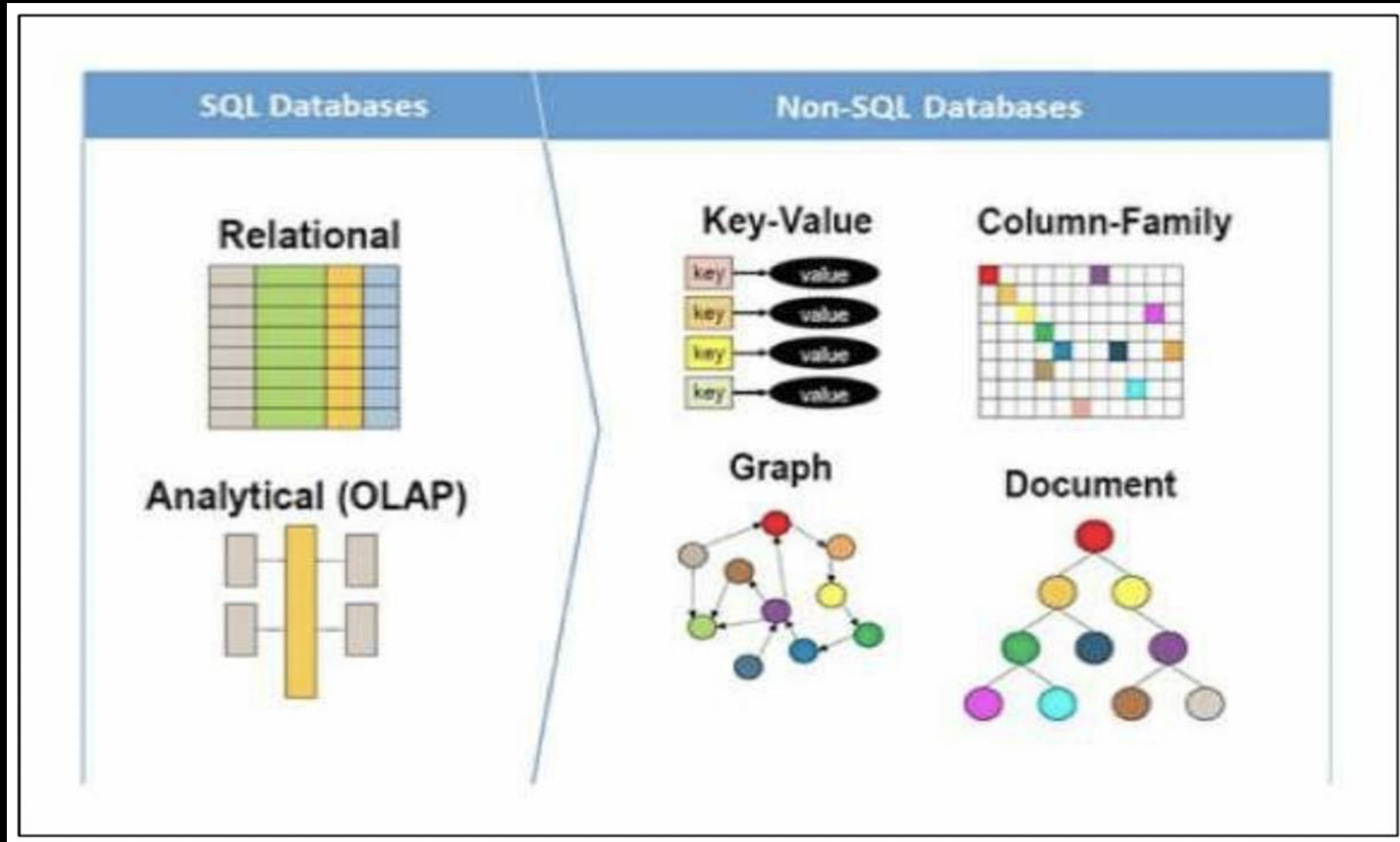
Comparativo: Barato x Rápido x Bom









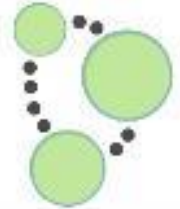

# Diferenças na operação



# Tipos de NoSQL



# Alguns exemplos de NoSQL

Type	Example	
Key-Value Store	 redis	 riak
Wide Column Store	 H·BASE	 cassandra
Document Store	 mongoDB	 CouchDB relax
Graph Store	 Neo4j	 The Distributed Graph Database

# Quando usar bancos NoSQL:

- ✓ Flexibilidade
- ✓ Escalabilidade
- ✓ Disponibilidade
- ✓ Baixo custo operacional
- ✓ Sem necessidade de schema
- ✓ Funcionalidades diferenciadas

Além disso...

- ✓ Facilitam integrações por eventos
- ✓ Consultas e gravações em sub-segundo
- ✓ APIs nativas
- ✓ Open-source

# Bancos colunares



## DB-Engines Ranking of Wide Column Stores

The DB-Engines Ranking ranks database management systems according to their popularity. The ranking is updated monthly.

This is a partial list of the [complete ranking](#) showing only wide column stores.

Read more about the [method](#) of calculating the scores.



☐ include secondary database models

13 systems in ranking, March 2023

Rank			DBMS	Database Model	Score		
Mar 2023	Feb 2023	Mar 2022			Mar 2023	Feb 2023	Mar 2022
1.	1.	1.	Cassandra +	Wide column	113.79	-2.43	-8.35
2.	2.	2.	HBase	Wide column	37.62	-0.79	-7.01
3.	3.	3.	Microsoft Azure Cosmos DB +	Multi-model i	36.10	-0.40	-4.79
4.	4.	4.	Datastax Enterprise +	Wide column, Multi-model i	7.33	-0.87	-2.56
5.	↑ 8.	↑ 7.	ScyllaDB +	Wide column, Multi-model i	5.76	+0.13	+1.77
6.	↓ 5.	↓ 5.	Microsoft Azure Table Storage	Wide column	5.73	-0.19	+0.09
7.	↓ 6.	↓ 6.	Google Cloud Bigtable	Multi-model i	5.44	-0.47	+1.10
8.	↓ 7.	8.	Accumulo	Wide column	5.39	-0.36	+1.45
9.	9.	9.	HPE Ezmeral Data Fabric	Multi-model i	1.17	-0.12	+0.35
10.	10.	10.	Amazon Keyspaces	Wide column	0.85	-0.06	+0.32

### Características:

- ✓ É possível agrupar famílias de dados para um mesmo ID
- ✓ A separação por família otimiza o tempo de consultas e gravações, melhora o particionamento
- ✓ Versionamento dos dados

### Usos:

- ✓ Aplicações com alto volume de consultas com usos diferentes para o mesmo ID (ex: cadastro do cliente)
- ✓ Necessidade de versionamento (ex: séries temporais)

# Benefícios e aplicações

- ✓ Velocidade de consulta e gravação de registros
- ✓ Registros com esquema flexível a cada família de coluna
- ✓ Usados para gravação logs, bases de atributos de objetos, IoT, Análise de dados em tempo real
- ✓ Demais dados que possuam uma chave de identificação e tenham um conteúdo de dados que seja variável

45697056



# Banco colunar: Cassandra

O Cassandra é um banco de dados NoSQL que pertence à categoria de banco de dados da column-store NoSQL.

Ele é um projeto Apache e possui uma versão Enterprise mantida pelo DataStax.

O Cassandra é escrito em Java e é usado principalmente para dados de séries temporais, como métricas, IoT (Internet of Things), logs, mensagens de bate-papo, e assim por diante. Ele é capaz de lidar com uma enorme quantidade de gravações e leituras e escala para milhares de nós.



45697056



# Banco colunar: Cassandra

## Alta escalabilidade

O Cassandra é altamente escalável, o que facilita a adição de mais hardware para conectar mais clientes e mais dados, conforme a necessidade;

## Arquitetura Rígida

O Cassandra não possui um único ponto de falha e está continuamente disponível para aplicativos essenciais aos negócios que não podem pagar por uma falha.

## Desempenho rápido em escala linear

Cassandra é linearmente escalável. Isso aumenta sua taxa de transferência porque facilita o aumento do número de nós no cluster. Por isso, mantém um tempo de resposta rápido;



# Banco colunar: Cassandra

## **Tolerante a falhas**

Cassandra é tolerante a falhas. Suponha que haja 4 nós em um cluster, aqui cada nó tem uma cópia dos mesmos dados. Se um nó não estiver mais em serviço, outros três nós poderão ser atendidos por solicitação;

## **Armazenamento de dados flexível**

O Cassandra suporta todos os formatos de dados possíveis, como estruturado, semiestruturado e não estruturado. Isso facilita você a fazer alterações em suas estruturas de dados de acordo com sua necessidade;

## **Distribuição Fácil de Dados**

A distribuição de dados no Cassandra é muito fácil, pois fornece a flexibilidade de distribuir dados onde você precisa, replicando dados em vários datacenters;



# Banco colunar: Cassandra

## **Suporte de Transação**

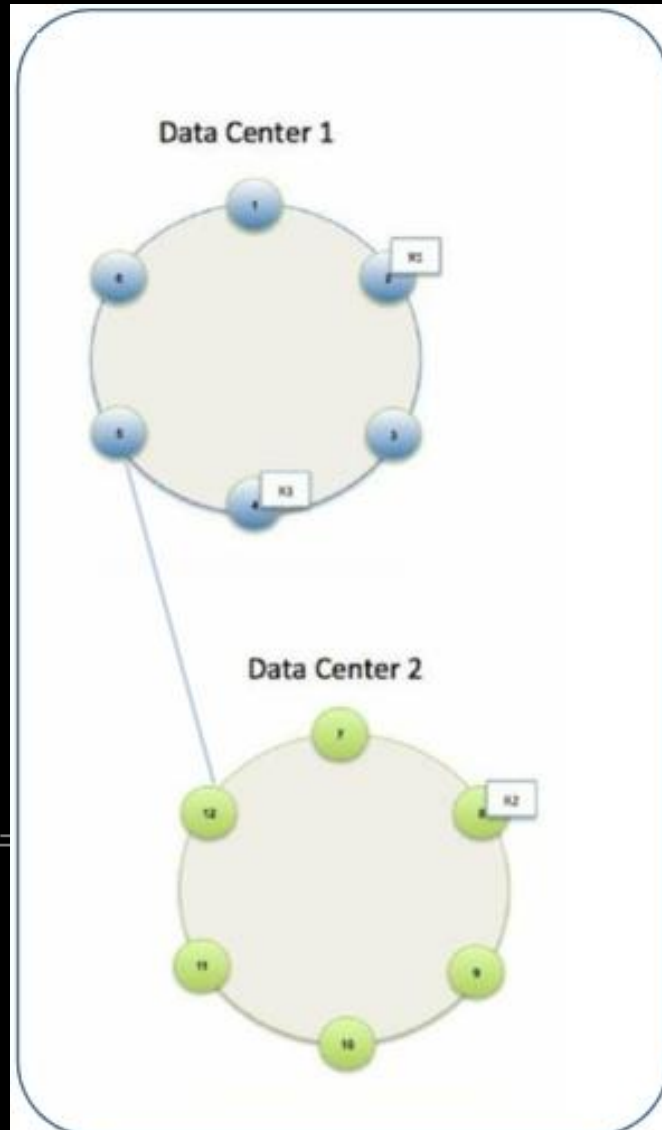
O Cassandra suporta propriedades como Atomicidade, Consistência, Isolamento e Durabilidade (ACID);

## **Escreve rápido**

O Cassandra foi projetado para rodar em hardware barato. Ele executa gravações incrivelmente rápidas e pode armazenar centenas de terabytes de dados, sem sacrificar a eficiência de leitura.



# Arquitetura Cassandra



## Componentes Cassandra:

**Nó** - é o local onde os dados são armazenados. É o componente básico de Cassandra.

**Data Center** - é uma coleção de nós. Muitos nós são categorizados como um data center.

**Cluster** - é a coleção de muitos data centers.



# Replicação de dados

Os dados são replicados para garantir nenhum ponto único de falha em caso de falha de hardware ou rede/conectividade, neste caso o Cassandra coloca réplicas de dados em diferentes nós com base nesses dois fatores:

Onde colocar a próxima réplica é determinado pela **Estratégia de Replicação**. Enquanto o número total de réplicas colocadas em nós diferentes é determinado pelo **Fator de Replicação**.

Um fator de replicação significa que há apenas uma única cópia de dados, enquanto três fatores de replicação significam que há três cópias dos dados em três nós diferentes. Para garantir que não haja um único ponto de falha, o fator de replicação deve ser três.



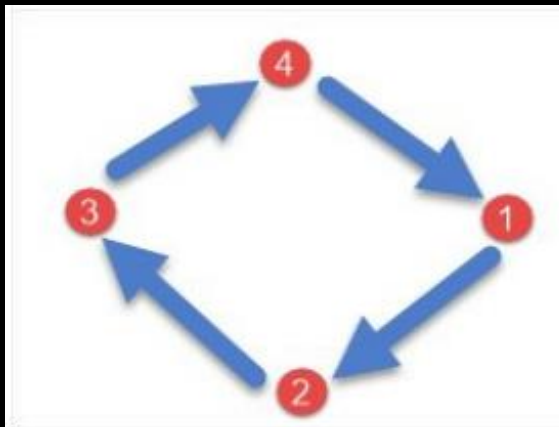
# Estratégia de replicação

Existem dois tipos de estratégias de replicação no Cassandra:

**SimpleStrategy**

**NetworkTopologyStrategy**

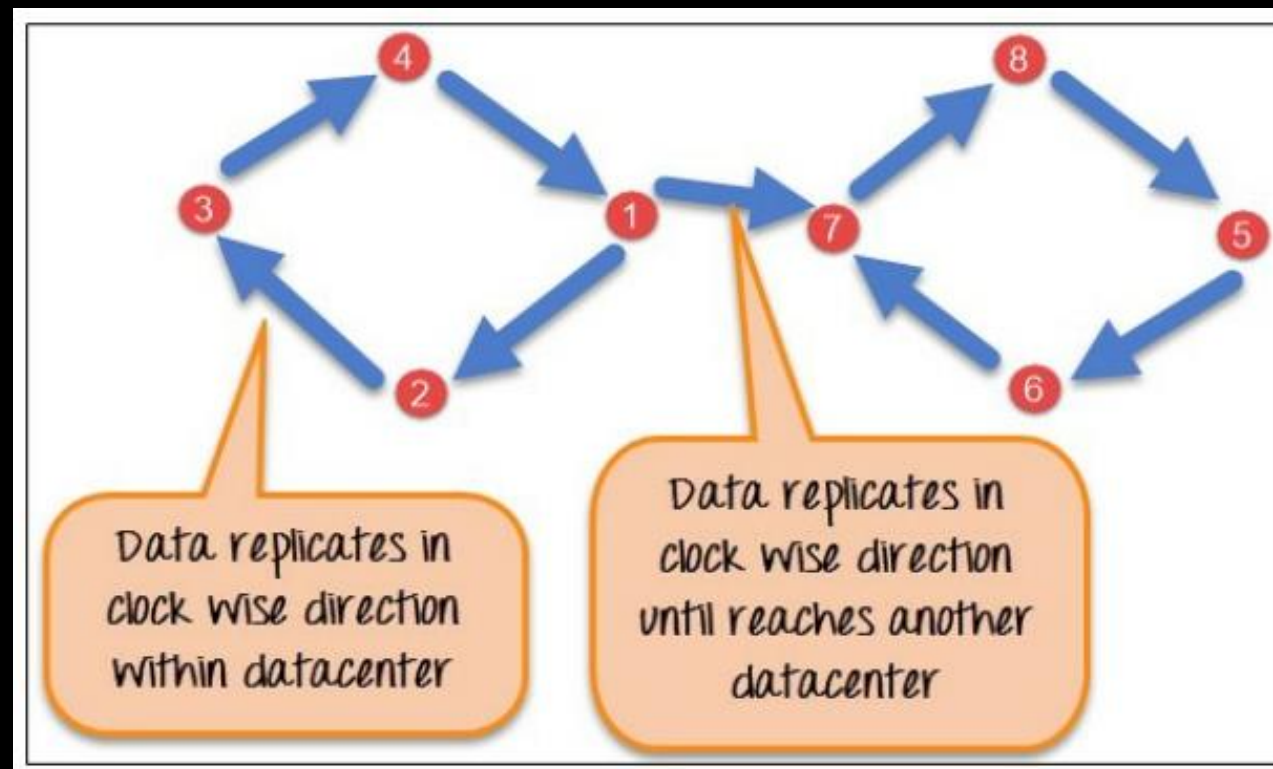
O **SimpleStrategy** é usado quando você tem apenas um data center. SimpleStrategy coloca a primeira réplica no nó selecionado pelo particionador. Depois disso, as réplicas restantes são colocadas no sentido horário no anel do nó.



# Estratégia de replicação

**NetworkTopologyStrategy** é usado quando você tem mais de dois data centers. As réplicas são definidas para cada data center separadamente. **NetworkTopologyStrategy** coloca réplicas no sentido horário no anel até atingir o primeiro nó em outro rack.

Essa estratégia tenta colocar réplicas em diferentes racks no mesmo data center. Isso se deve à razão de que, às vezes, falhas ou problemas podem ocorrer no rack. Em seguida, as réplicas em outros nós podem fornecer dados.





# Exercício:

Explorando um banco Cassandra

<https://cassandra.apache.org/>

<https://cassandra.apache.org/doc/latest/>





# Exercício Cassandra



O Docker já está instalado e uma imagem chamada **cassandra** já está disponível. Siga os passos iniciais abaixo:

1. Criar um Container a partir da imagem
2. Verifique o estado do cluster Cassandra
3. Acesse o CLI do Cassandra **cqlsh**
4. Crie um keyspace chamado **ksdts**
5. Acesse o keyspace **ksdts**
6. Crie a tabela **aula\_cassandra**
7. Insira um registro na tabela
8. Consulte a tabela (\*)
9. Inclua um novo campo
10. Consulte a tabela (\*)

Script disponibilizado  
com os comandos

Observaram algo de diferente na  
consulta?

Observaram algo de diferente na  
consulta?

45697056

# Continuação exercício Cassandra...

## Tipos de dados multivalorados

11. Criar a tabela **aula\_cassandra\_set**
12. Inserir um registro na tabela **aula\_cassandra\_set** (\*)
13. Consultar o registro inserido
14. Criar a tabela **aula\_cassandra\_list**
15. Inserir um registro na tabela **aula\_cassandra\_list** (\*)
16. Consultar o registro inserido
17. Criar a tabela **aula\_cassandra\_map**
18. Inserir um registro na tabela **aula\_cassandra\_map** (\*)
19. Consultar o registro inserido
20. Criar a tabela **aula\_cassandra\_tuple** +
21. Inserir um registro na tabela **aula\_cassandra\_tuple** (\*)
22. Consultar o registro inserido

Conjunto de coleções com um ou mais elementos ordenados

Conjunto com lista de elementos mantidos na ordem inserida

Conjunto com mapa de chave-valor

Conjunto capaz de armazenar conjuntos de dados. Aceita até 32768 campos

# Continuação exercício Cassandra...

## Índices e filtros

23. Inserir mais registros na tabela *aula\_cassandra*
24. Consultar campo\_texto na tabela *aula\_cassandra*
25. Fazer nova consulta com *allow filtering* (\*)
26. Criar um índice na tabela *aula\_cassandra* (\*)
27. Consultar o registro inserido usando índice
28. Inserir registros na tabela *aula\_cassandra\_map*
29. Criar índice map key
30. Consultar pela *chave* do campo map
31. Criar índice map value
32. Consultar pelo *valor* do campo map
33. Criar índice map entries
34. Consultar pela *entrada* do campo map

Em tabelas muito grandes, o tempo de consulta será elevado

A criação de índices é eficiente, mas pode aumentar os tempos de gravação

# Continuação exercício Cassandra...

## Exclusão de dados e tabelas

- 35. Exclua registros na tabela *aula\_cassandra*
- 36. Excluir dados de uma coluna na tabela *aula\_cassandra*
- 37. Excluir conteúdo campo set na *aula\_cassandra\_set*
- 38. Excluir conteúdo campo list na *aula\_cassandra\_list* (\*)
- 39. Excluir conteúdo campo map na *aula\_cassandra\_map*
- 40. Faça um truncate na tabela *aula\_cassandra\_list*
- 41. Excluir um dos índices da tabela *aula\_cassandra\_map*
- 42. Excluir a tabela *aula\_cassandra\_set*
- 43. Criar um keyspace e excluí-lo para testar

O índice na lista inicia com 0

# Desafios

1. O que acontece se inserir o mesmo valor duas vezes?
2. O que acontece se inserir a mesma chave, mas com valores diferentes?
3. Atualize registros nas tabelas
4. É possível fazer delete sem usar a primary key?
5. Faça updates nos campos multivalorados set, list e map
6. Inserir registros no campo set fora de ordem
7. Criar tabela com campos tuple aninhados

# Questão

**Quais foram as diferenças entre fazer estes exercício usando um banco relacional e usando um banco colunar?**

  
**MBA<sup>+</sup>**

Copyright © **2023** Profs. Ivan Gancev e Leandro Mendes

Todos direitos reservados. Reprodução ou divulgação total ou parcial deste documento é expressamente proibido sem o consentimento formal, por escrito, dos Professores Ivan Gancev e Leandro Mendes

[profivan.gancev@fiap.com.br](mailto:profivan.gancev@fiap.com.br)

[profleandro.mendes@fiap.com.br](mailto:profleandro.mendes@fiap.com.br)

