

Armazenamento de Dados

Capítulo 01 – Banco de Dados Relacionais

PROF.: RICARDO BRITO ALVES

Banco de Dados Relacionais

Capítulo 01 – Aula 01.01 – Introdução a Banco de Dados

PROF.: RICARDO BRITO ALVES

Nesta Aula



- Banco de dados
- Aplicações de Banco de Dados
- Sistemas de Arquivos x Banco de Dados
- SGBD

Banco de Dados



O que é um **Banco de Dados**?

- Coleção de dados relacionados.
- Fatos conhecidos que podem ser registrados e possuem significado implícito.
- Representa algum aspecto do mundo real.
- Coleção logicamente coerente de dados com algum significado inherente.
- Construído para uma finalidade específica.

Bancos de dados possuem alguma *ligação com o mundo real, com a fonte que gera as informações, usuários*, algum ator que futuramente possa necessitar das informações armazenadas no banco de dados.

Aplicações de BD

Dados fazem parte do nosso dia-a-dia:

- Operação bancária.
- Reserva de hotel.
- Matrícula em uma disciplina da universidade.
- Cadastro na vídeo locadora.

Dado: fato do mundo real que está registrado - exemplos:

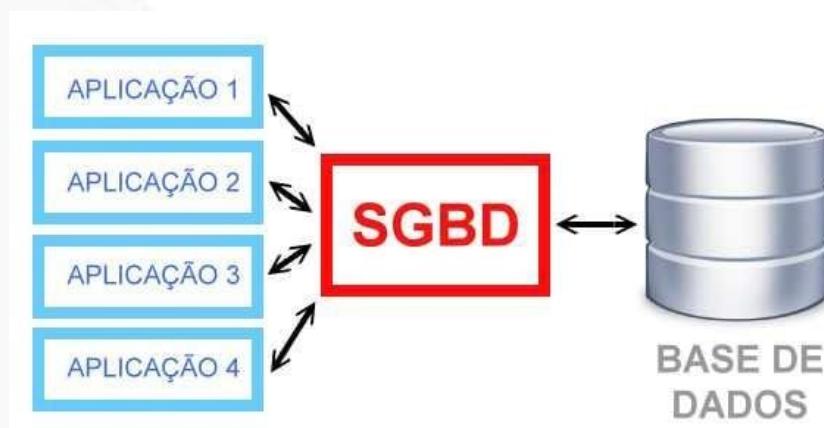
endereço, data.

Informação: fato útil que pode ser extraído direta ou indiretamente a partir dos dados - exemplos: endereço de entrega, idade.

Banco de Dados (BD): coleção de dados inter-relacionados e persistentes que representa um sub-conjunto dos fatos presentes em um domínio de aplicação (universo de discurso).

Aplicações de BD

- Sistemas de data warehousing e de processamento analítico on-line (OLAP)
 - Extrair e analisar informações comerciais úteis de bancos de dados muito grandes.
 - Ajuda na tomada de decisão.
- Tecnologia de tempo real e banco de dados ativo
 - Controla processos industriais e de manufatura.



Aplicações de BD



Banco de dados = instância de dado + meta-dados

- Instância de dado
 - Dado propriamente.

Meta-dados

- Dicionário de dados
 - Esquema da base de dados.
 - Acessado através de linguagens de definição de dados.

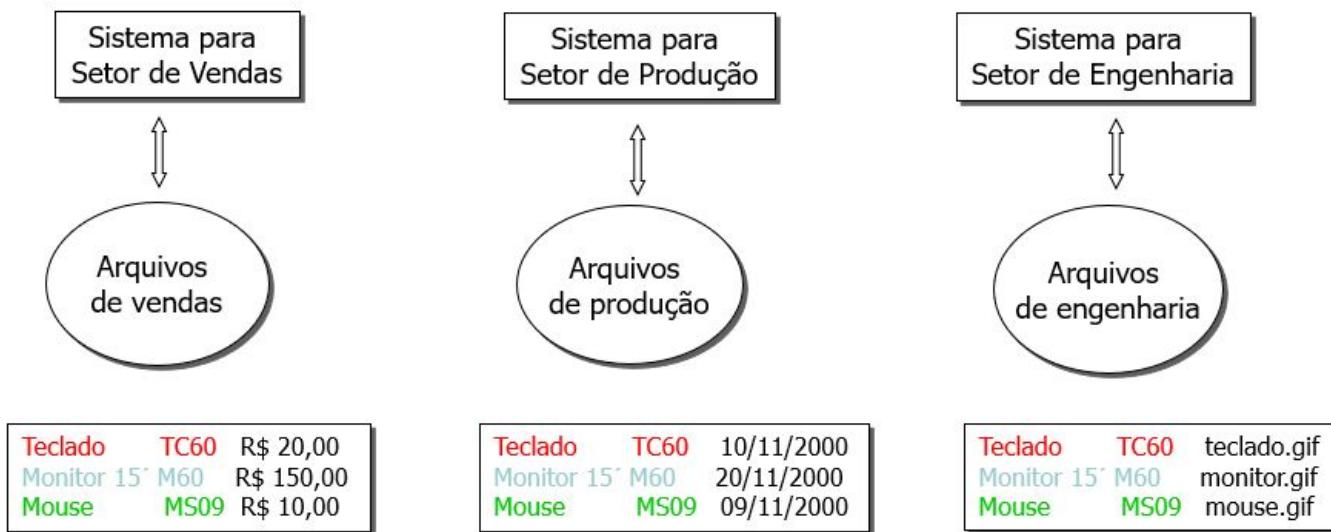
Sistemas de Arquivos



- Um sistema de arquivo é um conjunto de estruturas lógicas que *permite ao sistema operacional* ter acesso e controlar os dados gravados no disco.
- Cada sistema de arquivos possui as suas *peculiaridades, como limitações, qualidade, velocidade, gerenciamento de espaço*, entre outras características.
- *O sistema de arquivos é que define como os bytes que compõem um arquivo serão armazenados no disco e de que forma o sistema operacional terá acesso aos dados.*

Sistemas de Arquivos

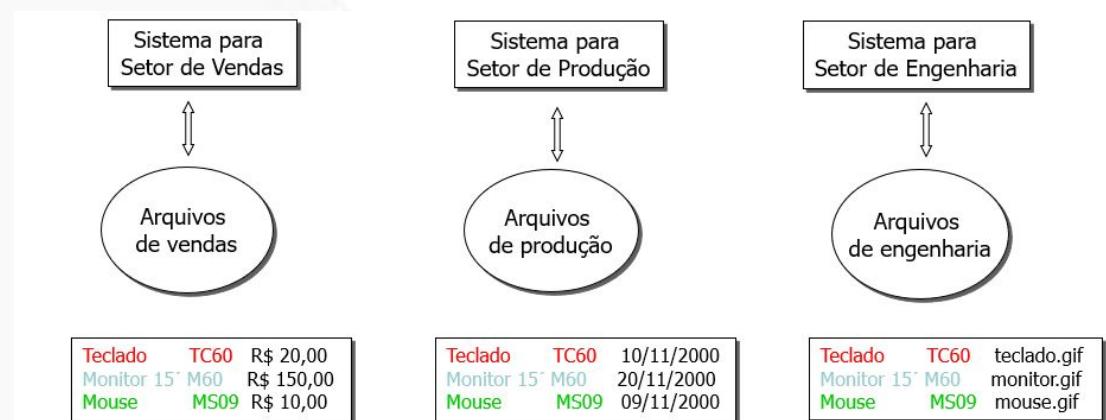
- Mesmo objeto da realidade é representado várias vezes na base de dados
 - Exemplo - teclado, monitor e mouse



Sistemas de Arquivos

IGTI

- Redundância não controlada de dados
 - Não há gestão automática da redundância
 - Redundância leva a:
 - Inconsistência dos dados
 - Re-digitação de informações
 - Dificuldade de extração de informações
- Dados pouco confiáveis e de baixa disponibilidade



Sistemas de Arquivos



- Concorrência
 - Difícil implementação.
 - Políticas de acesso concorrente consistente são independentes de domínio.
- Tolerância a falhas
 - Falta de luz, erro de disco, interrupção de funcionamento, etc.
 - Cópias? restauração do estado anterior? Consistência da base?
- Segurança
 - Acesso diferenciado por tipo de usuário.

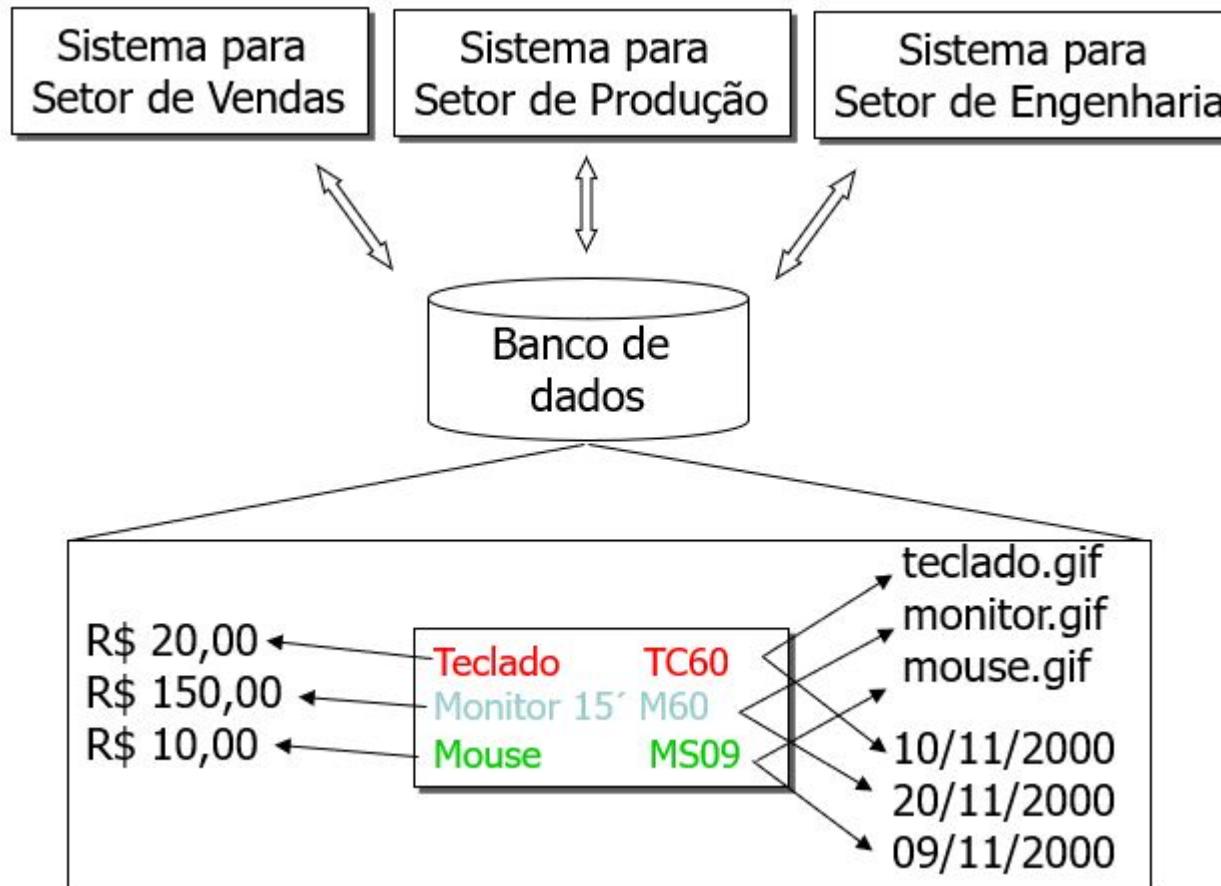
Sistemas de Arquivos



- Outros problemas:
 - Número máximo de arquivos.
 - Tamanho de memória.
 - Limitações do tipo de arquivo, tipo de acesso.
 - Preocupações técnicas junto com problemas do domínio.
- Exemplo: efetuar aluguel de um DVD
 - Sem reservas? sem multas?
 - Como registrar um empréstimo?
 - Abrir arquivos (fechando outros ...).
 - Carregar registros na memória (abre índice, usa ponteiro, estourou memória?,).

Banco de Dados

IGTI



SGBD



- SGBD é um conjunto de programas e ferramentas utilizadas para configurar, atualizar e manter um banco de dados e serve para armazenar de forma organizada as Informações.
- SGBD é um sistema que é projetado para gerir os volumes de informações que são armazenados em um banco de dados.

- ✓ Recursos para administrar usuários/permissões.
- ✓ Recursos para criar/alterar tabelas e banco de dados.
- ✓ Recursos para backup e restauração de dados.
- ✓ Recursos para otimizar a performance do banco.

Principais SGBDs

IGTI



www.oracle.com



www.firebirdsql.org/



www.microsoft.com/sqlserver/



www.mysql.com/



www-01.ibm.com/software/data/db2/



www.sybase.com.br/



www.postgresql.org/

SGBD

IGTi



Finalidade dos SGBDs

- Reduzir a redundância e inconsistência nos dados
 - ✓ Garantir que uma determinada informação esteja armazenada em apenas um local.
 - ✓ Alterações nos dados deverão ser controladas e automaticamente propagadas à todos os usuários que acessam a informação.
- Reduzir a dificuldade no acesso aos dados
 - ✓ Os SGBD's fornecem linguagens e mecanismos eficientes para que uma determinada informação seja encontrada.
- Problemas de segurança
 - ✓ Os SGBD's possuem usuários e perfis de acesso, de modo que nem todos os usuários tem acesso a todas as informações.

Finalidade dos SGBDs

- Garantir o isolamento dos dados
 - ✓ Manter todos os dados gravados no banco de dados com o mesmo tipo de formato, bem como impedir que o dados seja acessado de outro local que não seja o SGBD, garantindo, assim, um único ponto de acesso ao dado.
- Minimizar problemas de integridade
 - ✓ Algumas informações de uma empresa devem seguir algumas regras ou restrições, e o SGBD provê formas mais fáceis de garantir a integridade das informações através dessas restrições.
- Resolver problemas de atomicidade
 - ✓ O SGBD deve garantir todas as operações atômicas, ou seja, todas as operações ou transações que devem acontecer completamente ou devem ser desfeitas caso ocorram falhas, são gerenciadas pelo SGBD.

Conclusão



- ✓ Banco de dados
- ✓ Aplicações de Banco de Dados
- ✓ Sistemas de Arquivos x Banco de Dados
- ✓ SGBD

Próxima Aula



01. ••

Banco de Dados Relacional

02. ••

03. ••

04. ••

Banco de Dados Relacionais

Capítulo 01 – Aula 01.02 – Banco de Dados Relacionais

PROF.: RICARDO BRITO ALVES

Nesta Aula



- ❑ Bancos de Dados Relacional
- ❑ Princípios do Modelo Relacional
- ❑ Composição do Modelo Relacional
- ❑ Benefícios Banco Relacional
- ❑ Possíveis razões para usar um Banco Relacional
- ❑ Componentes do Banco de Dados Relacional

Banco de Dados Relacional



- Sistema Gerenciador de Banco de Dados Relacional (SGBDR) ou Relational database management system (RDBMS).
- É um tipo de banco de dados que armazena e fornece acesso a pontos de dados relacionados entre si.
- São baseados no modelo relacional, uma maneira intuitiva e direta de representar dados em tabelas.
- Cada linha na tabela é um registro com uma ID exclusiva chamada chave.
- As colunas da tabela contêm atributos dos dados e cada registro geralmente tem um valor para cada atributo, facilitando o estabelecimento das relações entre os pontos de dados.

Composição do Modelo Relacional

O modelo relacional é composto, basicamente, pelos seguintes elementos:

- *Coleções de objetos ou relações* que armazenam os dados.
- Um conjunto de *operadores* que agem nas relações, produzindo outras relações.
- *Integridade de dados, para precisão e consistência.*

Benefícios dos Bancos de Dados Relacionais



- ✓ É usado por organizações de todos os *tipos e tamanhos* para uma ampla variedade de necessidades de informações.
- ✓ Pode ser considerado para qualquer necessidade de informações na qual *os pontos de dados se relacionam entre si e devem ser gerenciados de maneira segura e consistente, com base em regras*.
- ✓ O modelo relacional é o modelo mais amplamente utilizado para bancos de dados.

Possíveis razões para usar um Banco Relacional



- ✓ Seus dados são estruturados e imutáveis.
- ✓ Possui quatro propriedades essenciais que definem as transações do banco de dados – ACID, que reduz possíveis anomalias e protege a integridade.
 - **A**tomicidade garante que as transações sejam atômicas (indivisíveis). A transação será executada totalmente ou não será executada.
 - **C**onsistência garante que o banco de dados passará de uma forma consistente para outra forma consistente.
 - **I**solamento garante que a transação não será interferida por nenhuma outra transação concorrente.
 - **D**urabilidade garante que o que foi salvo, não será mais perdido.

Transações

- ✓ Uma transação é uma unidade lógica de processamento no banco de dados. Uma transação inclui uma ou mais operações de acesso ao banco de dados e englobam operações de inserção, exclusão, alteração ou recuperação.
- ✓ Por que a Restauração (Recuperação) é necessária? O sistema deverá garantir que:
 - ✓ Todas as operações na transação foram completadas com sucesso e seu efeito será gravado permanentemente no banco de dados.
 - ✓ A transação não terá nenhum efeito sobre o banco de dados ou sobre quaisquer outras transações.

Componentes do Banco de Dados Relacional

Os principais tipos de objetos que fazem parte de um banco de dados relacional são:

- **Tabela**: estrutura básica de armazenamento no SGBDR.
- **Tupla ou Linha / Registro**: representa todos os dados requeridos de uma entidade em particular. Cada linha em uma tabela deve ser identificada por uma chave primária, de modo a não haver duplicação de registros.
- **Coluna ou Campo**: Unidade que armazena um tipo específico de dado (valor) – ou não armazena nada, com valor nulo. Esta é uma coluna não-chave, significando que seu valor pode se repetir em outras linhas da tabela.

Componentes do Banco de Dados Relacional

- **Relacionamento:** Associação entre as entidades (tabelas), conectadas por chaves primárias e chaves estrangeiras.
- **Chave Primária (Primary Key / PK)**: coluna (atributo) que identifica um registro de forma exclusiva na tabela.
- **Chave Estrangeira (Foreign Key / FK)**: coluna que define como as tabelas se relacionam umas com as outras. Uma FK se refere a uma PK ou a uma chave única em outra tabela (ou na mesma tabela!).
- **Restrições (Constraints)**: Propriedades específicas de determinadas colunas de uma relação, se a coluna pode aceitar valores nulos ou não.
- **Outros: Índices, Stored Procedures, Triggers, etc.**

Conclusão



- ✓ Banco de Dados Relacional
- ✓ Modelo Relacional

Próxima Aula



01. ••

Modelo de Dados

02. ••

03. ••

04. ••

Banco de Dados Relacionais

Capítulo 01 – Aula 01.03 – Introdução a Modelagem de Dados

PROF.: RICARDO BRITO ALVES

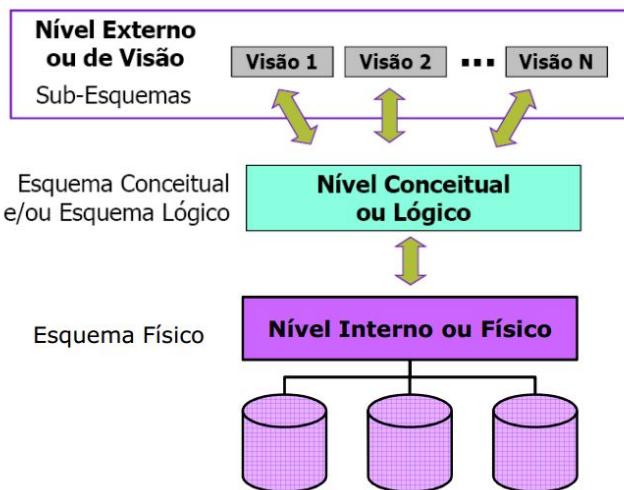
Nesta Aula



- Abstração de dados
- Modelo de Dados
- Tipos de Modelos de Dados

Abstração de Dados

- ✓ Visão abstrata do banco de dados, ou seja, para o usuário pouco importa qual unidade de armazenamento está sendo usada para guardar seus dados, contanto que os dados estejam disponíveis no momento necessário.
- ✓ Existem, portanto, maneiras diferentes de abstrair os dados, de forma que todos os usuários possam compreender o banco de dados.

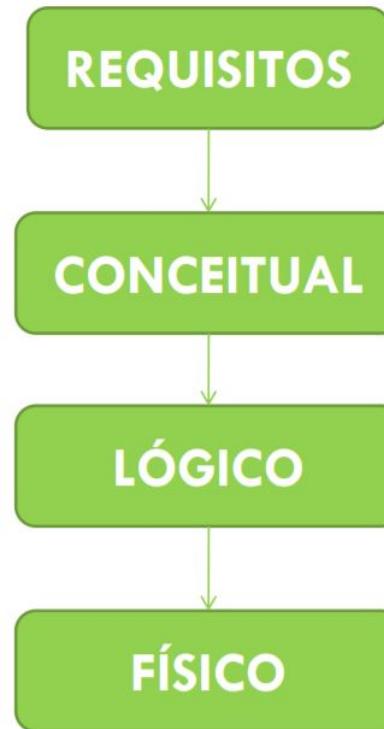


Abstração de Dados

Nível Conceitual: descreve apenas parte do banco de dados de forma simplificada. *Visão para o cliente.*

Nível Lógico: nível intermediário onde é possível representar todo o banco de dados, com suas estruturas e relacionamentos, mas sem se preocupar como o dado será gravado em disco. *Independe do SGBD, por exemplo.*

Nível Físico: é o nível de abstração mais baixo, ou seja, *demonstra como os dados serão armazenados fisicamente em disco.* Criação dos scripts, modelo físico, estratégias de segurança e armazenamento, etc.



Modelagem de Dados

Para uma modelagem de dados alcançar os objetivos esperados, ela deve fornecer ao desenvolvedor.

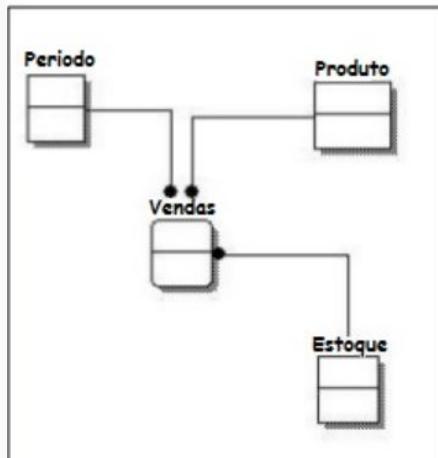
- Representar o ambiente.
- Documentar e normalizar.
- Fornecer processos de validação.
- Observar processos de relacionamentos entre objetos.

Modelo de Dados

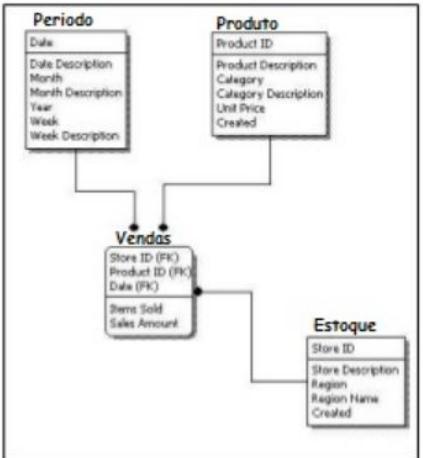
IGTI

O *modelo de dados* é a forma pela qual descrevemos o projeto de *bancos de dados*. O modelo de dados representa todos os dados, a forma como os dados se relacionam e as possíveis restrições sobre eles.

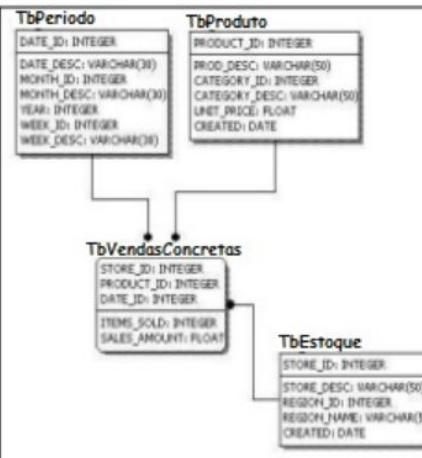
Tipos de modelos de dados



Modelo Conceitual



Modelo Lógico



Modelo Físico

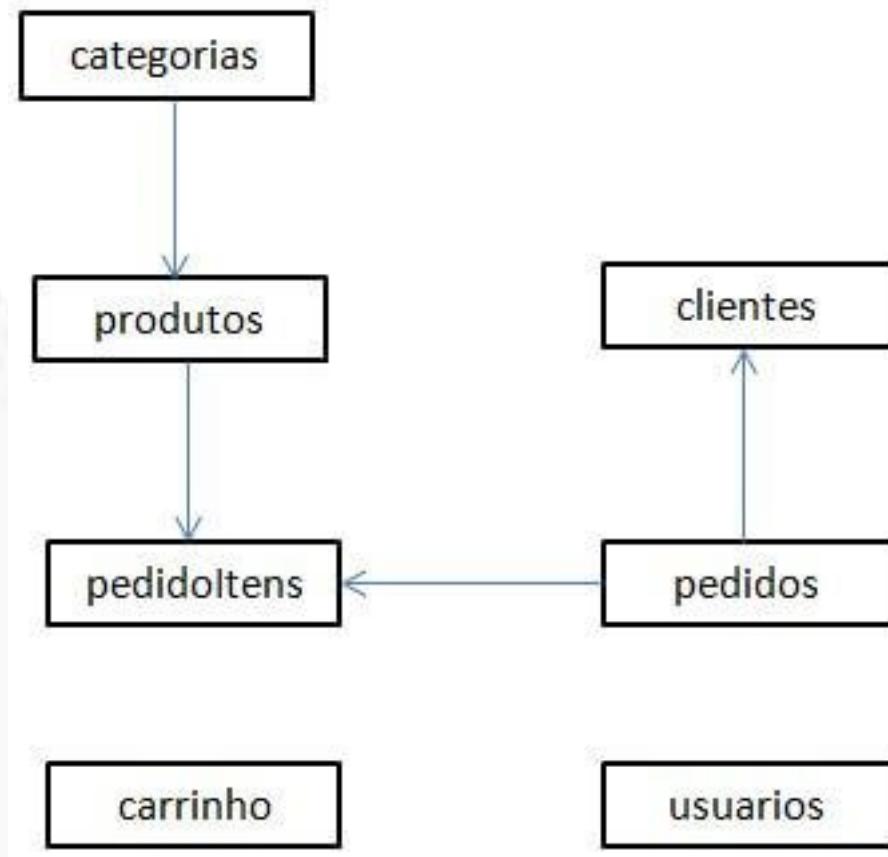
Modelo Conceitual



- *Primeira fase da modelagem, serve para representar em alto nível as entidades e seus relacionamentos em um nível macro*, considerando-se exclusivamente o ponto de vista do usuário gestor dos dados.
- Principal finalidade é capturar os requisitos de informação e regras de negócio sob o ponto de vista do negócio.
- *É independente de hardware ou software, ou seja, não depende de nenhum tipo de servidor de banco de dados.*

Modelo Conceitual

IGTI



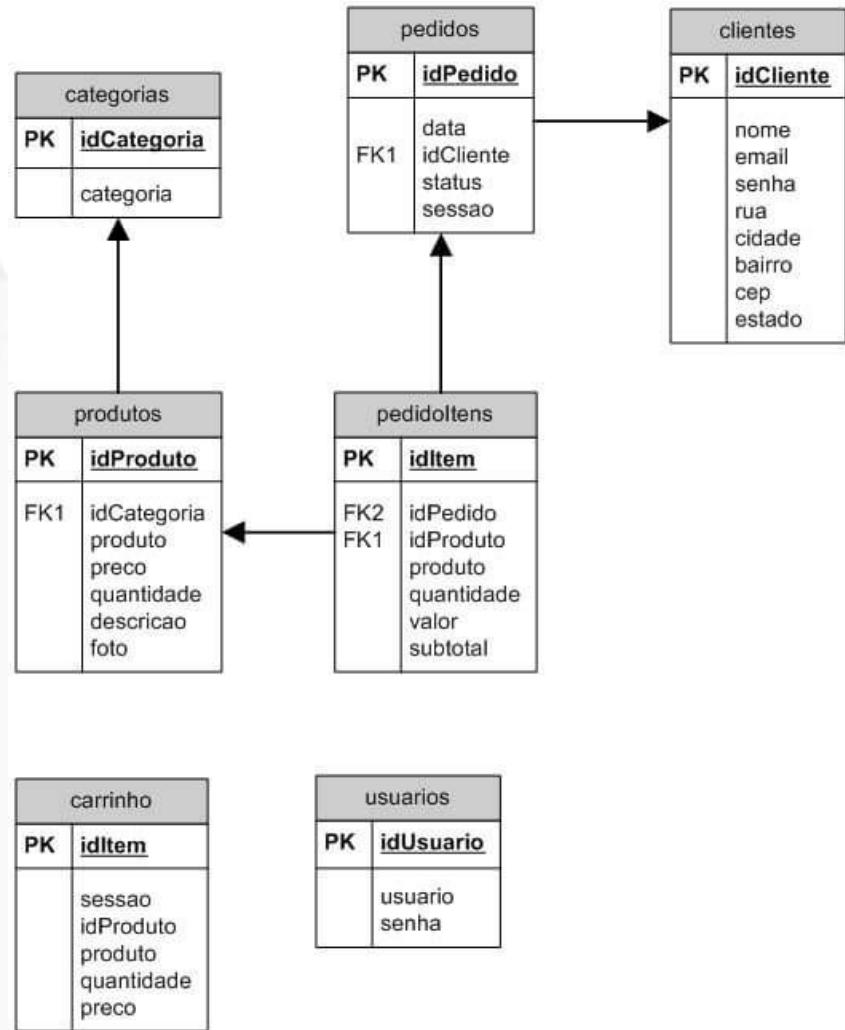
Modelo Lógico



- *Consiste em determinar quais informações serão necessárias ao banco de dados, divididas em Tabelas.*
- Também serão definidos nesta fase *os campos das Tabelas, seus atributos e propriedades e ainda as Chaves Primárias e Secundárias, seus relacionamentos, normalização, integridade referencial, entre outras.*
- Descreve como os dados serão armazenados no banco e também seus relacionamentos, *mas ainda são independentes do SGBD.*
- Modelagem lógica *não inclui índices e constraints.*

Modelo Lógico

IGTI

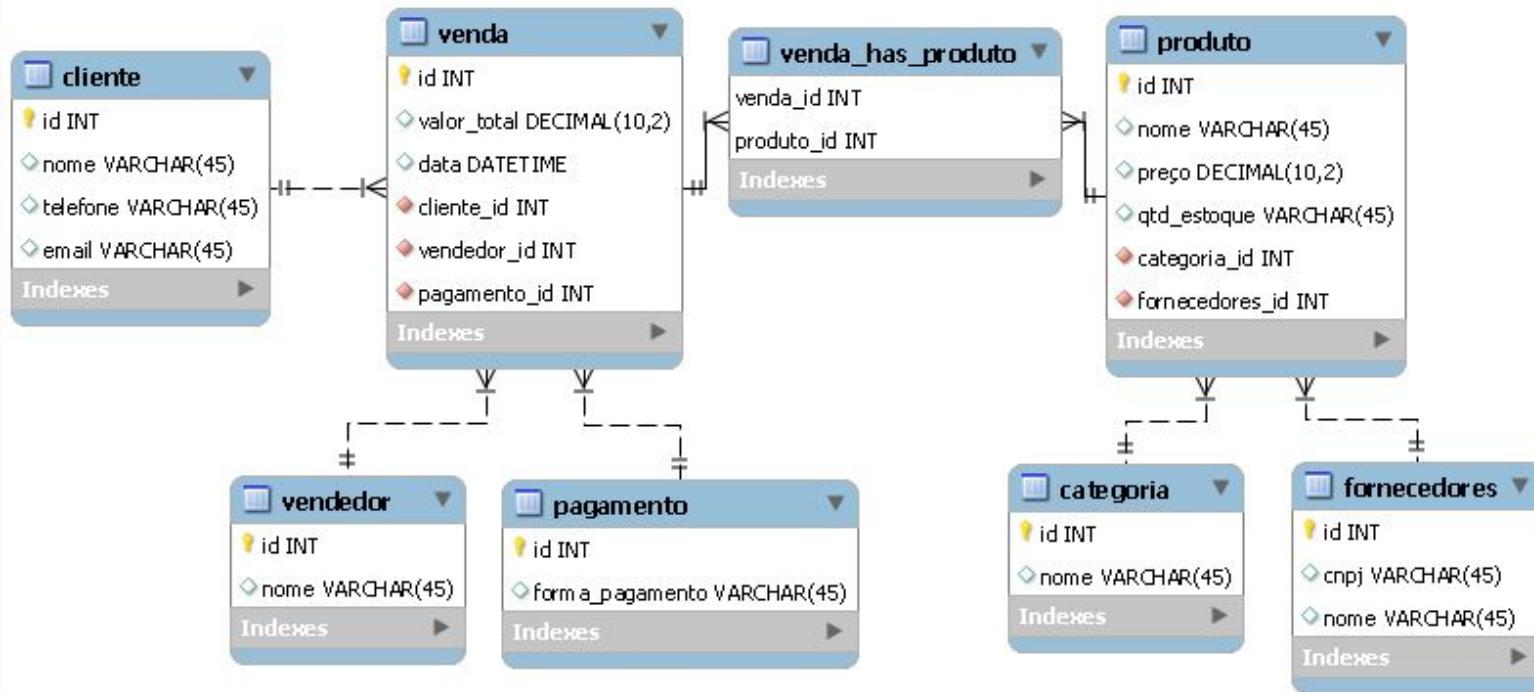


Modelo Físico

- ***Consiste na escolha de um SGBD.***
- Parte do Modelo de Entidades e Relacionamentos (MER) que é um modelo abstrato cuja finalidade é descrever, de maneira conceitual, os dados a serem utilizados em um sistema de informações ou que pertencem a um domínio.
- A principal ferramenta do modelo é sua representação gráfica, o Diagrama Entidade Relacionamento – DER.

Modelo Físico

IGTI



MER x DER



MER: Conjunto de conceitos e elementos de modelagem que o projetista de banco de dados precisa conhecer.
Modelo lógico.

DER: resultado do processo de modelagem executado pelo projetista de dados que conhece o MER. Modelo físico.

Tarefa para modelagem

- ✓ Identificar entidades.
- ✓ Identificar atributos.
- ✓ Identificar relacionamentos.
- ✓ Criar e associar chaves.
- ✓ Normalizar para reduzir redundância.
- ✓ Desnormalizar para aumentar performance.

Conclusão



- ✓ Abstração de dados
- ✓ Modelo de Dados e seus tipos

Próxima Aula



01. ••

Modelagem de Dados

02. ••

03. ••

04. ••

Armazenamento de Dados

Capítulo 01 – Aula 01.04 – Modelagem de Dados

PROF.: RICARDO BRITO ALVES

Nesta Aula



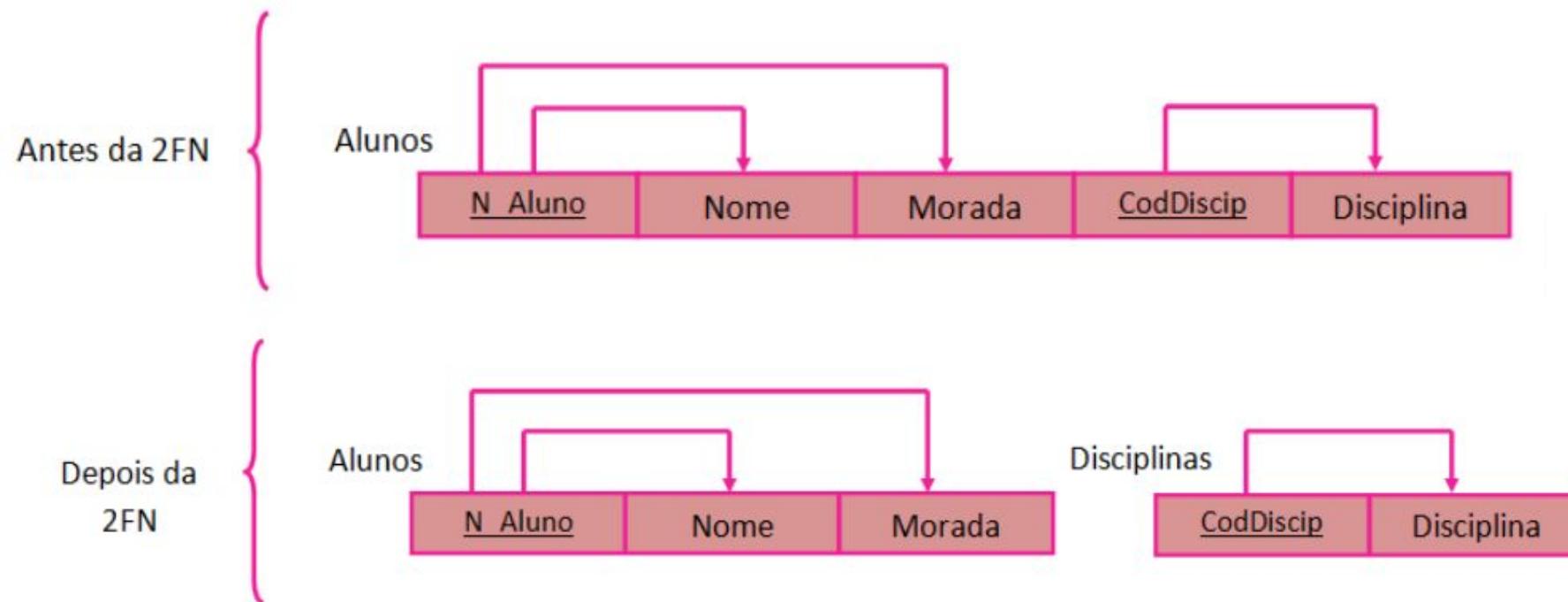
- Normalização
- Entidade
- Atributos
- Relacionamentos

Normalização

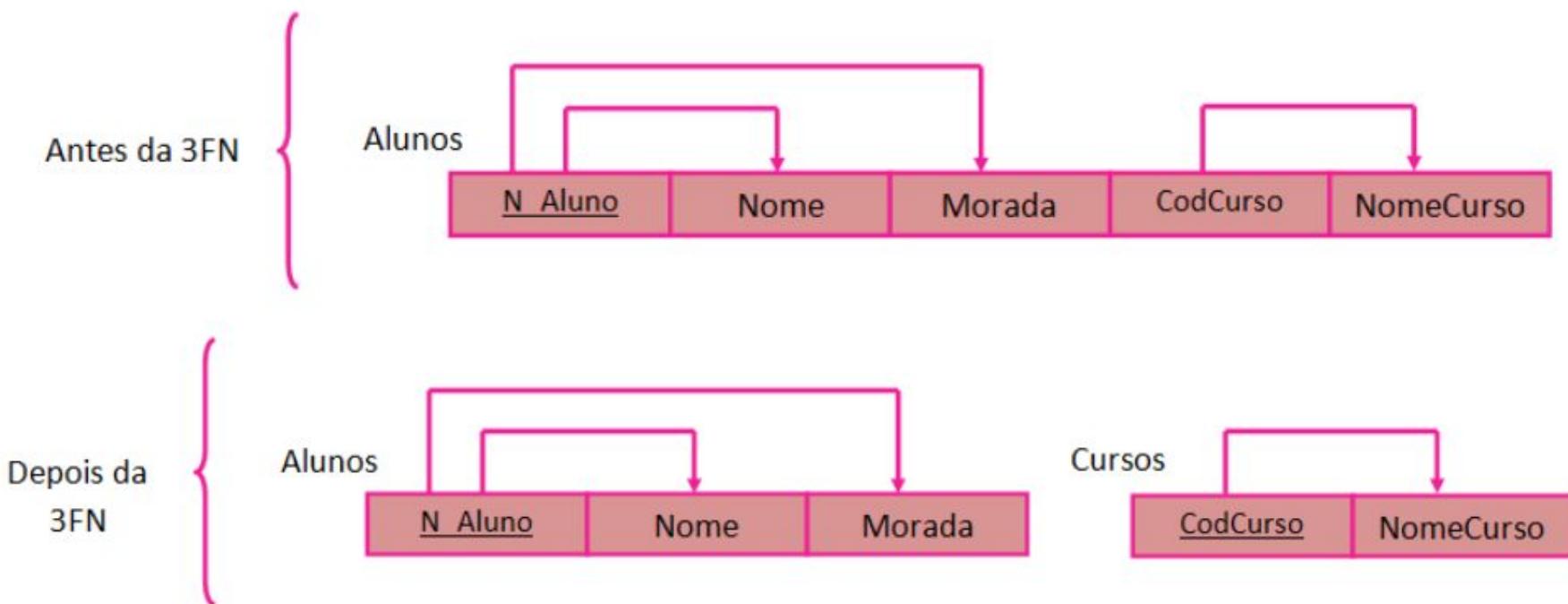
CodCliente	Cliente	Morada	Encomendas			
			N_Enc	Data	Produto	Quantidade
C01	Aníbal	Lisboa	1	2010-05-25	Ananás	10
			5	2010-05-30	Cebolas	20
C02	Belmiro	Braga	3	2010-05-26	Bananas	30

CodCliente	Cliente	Morada	N_Enc	Data	Produto	Quantidade
C01	Aníbal	Lisboa	1	2010-05-25	Ananás	10
C01	Aníbal	Lisboa	5	2010-05-30	Cebolas	20
C02	Belmiro	Braga	3	2010-05-26	Bananas	30
C03	Casimiro	Coimbra	2	2010-05-25	Tomates	50
C03	Casimiro	Coimbra	4	2010-05-26	Cebolas	20

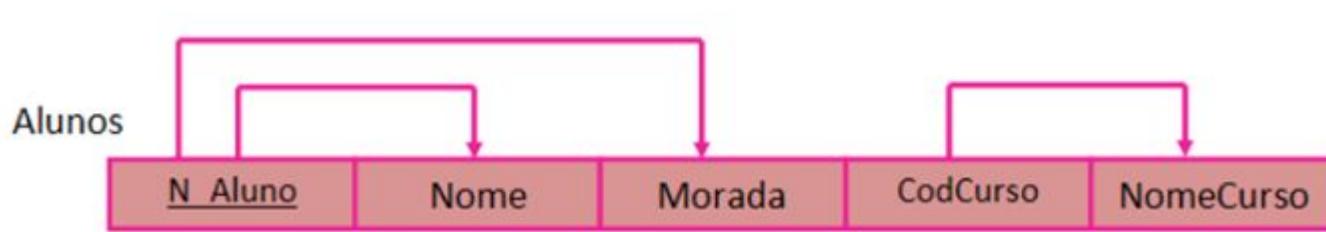
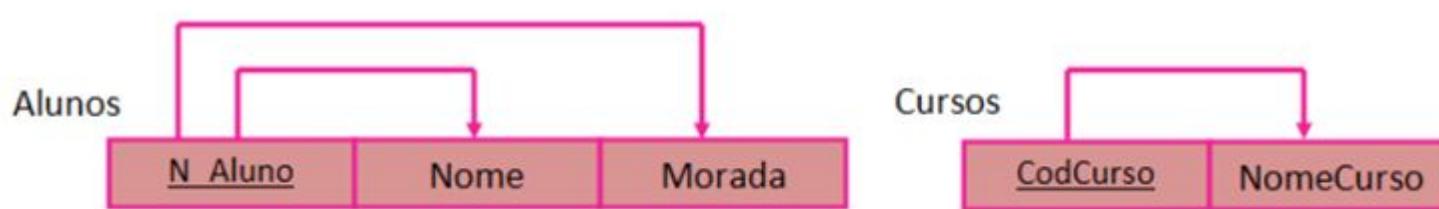
Normalização



Normalização



Desnormalização



Entidade

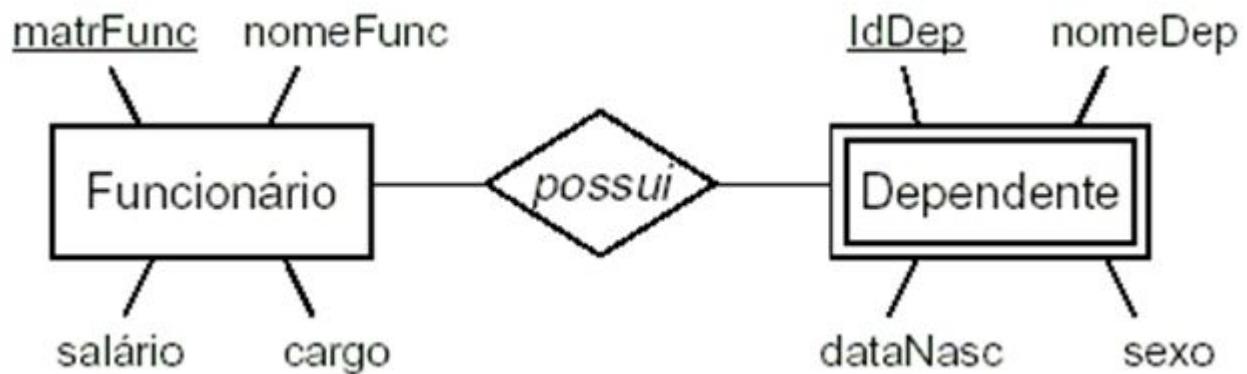
É uma representação concreta ou abstrata de um objeto, com características semelhantes, do mundo real. Ex.: Fornecedor, Pessoa, Imóvel, Curso.

EMPREGADO

Entidade Forte x Fraca

Entidade Fraca não existe se não estiver relacionada a outra, isto é, ela é logicamente dependente da outra.

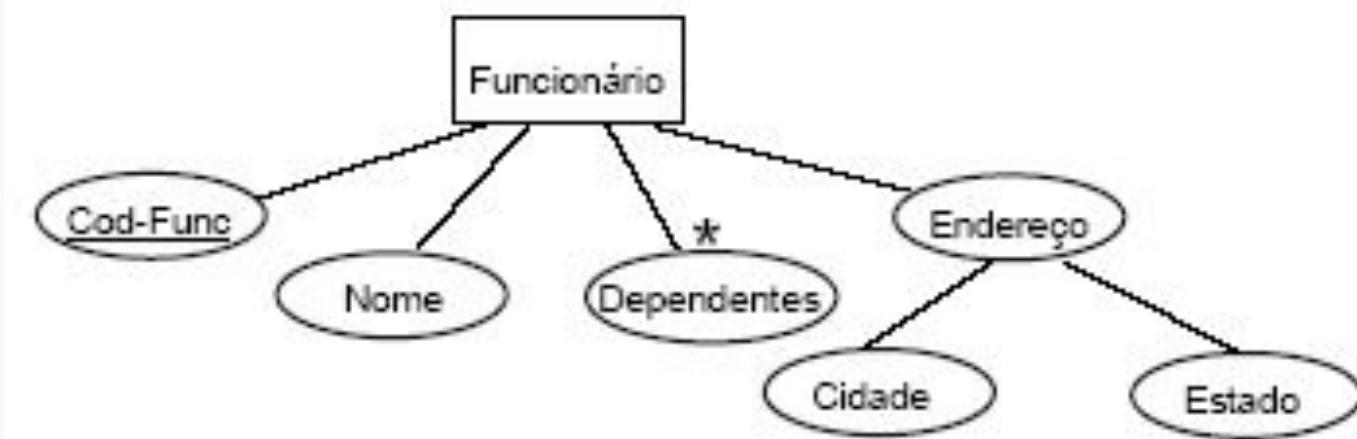
Alguns conjuntos entidade não possuem um conjunto de atributos capaz de identificar uma determinada entidade. Neste caso, sua existência depende da existência de outra entidade.



Entidade Forte x Fraca

IGTI

Elemento de dado que contém o valor de uma propriedade de uma entidade.



Atributo - Classificação



- Atributo Composto
- Atributo Não Único
- Atributo Obrigatório
- Atributo Multivvalorado

Atributo

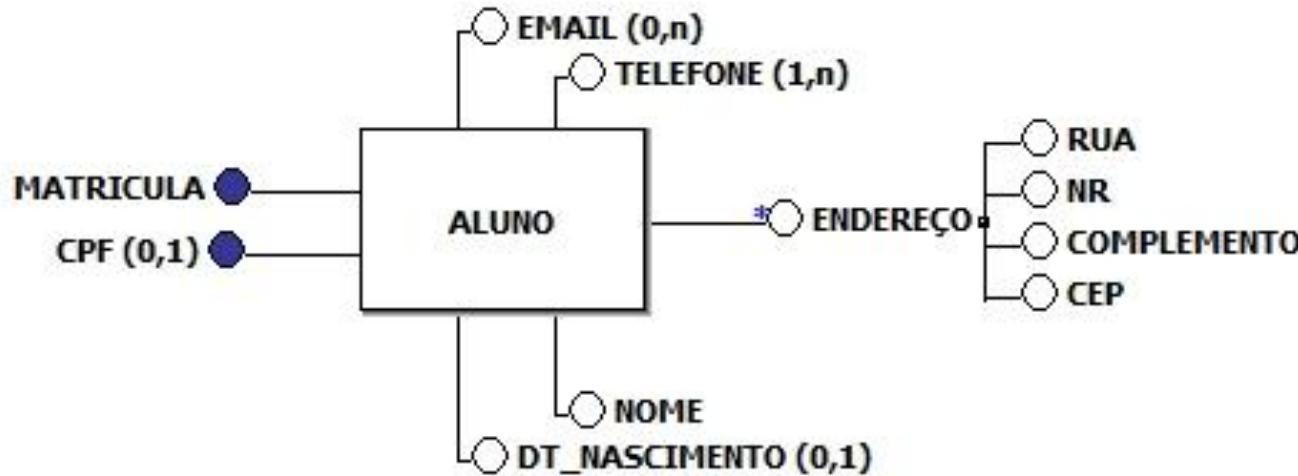
Atributo Identificador: identifica unicamente cada entidade de um conjunto-entidade, devem ser obrigatórios e únicos Ex.: Cod_Func.

Atributo Derivado: o seu valor pode ser calculado a partir do valor de outro(s) atributo(s). Ex.: idade (pode ser calculada a partir da data de nascimento).

Atributo

Domínio de um atributo: descrição de possíveis valores permitidos para um atributo. Ex.: Sexo {M, F}.

Tipo de um Atributo: determina a natureza dos valores permitidos para um atributo. Ex.: inteiro, real, string, etc.



Esquema x Instância

Esquema de um Banco de Dados é a especificação da estrutura do Banco de Dados.

Instância é o conjunto de ocorrências dos objetos de dados de um esquema em um dado momento do tempo.

Código	Nome	Sigla	
1	Tecnologia da Informação	TI	
2	Recursos Humanos	RH	

Relacionamentos

As entidades são conectadas umas às outras através de relacionamentos.

Ex.: As pessoas **Moram** em Apartamentos.

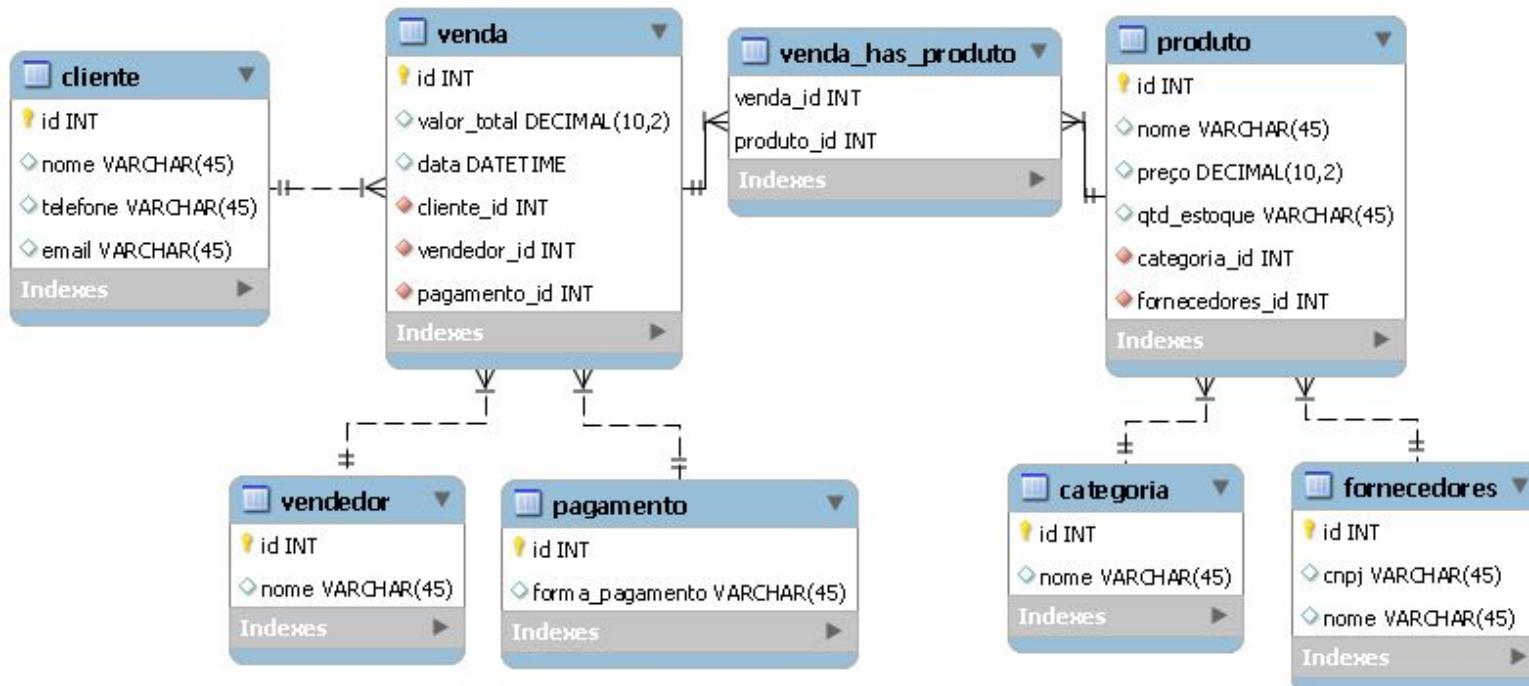
Os apartamentos **Formam** Condomínios.

Os condomínios **Localizam-se** em Ruas ou Avenidas.

As Avenidas e Ruas **Pertencem** a uma Cidade.

Relacionamentos

IGTI



Relacionamentos



Cardinalidade do relacionamento ($n = \text{vários}$)

1:1 (um para um - uma linha de uma tabela têm apenas um relacionamento com outra linha de outra tabela).

Um aluno mora atualmente em um único endereço).

1:N (um para n - uma linha de uma tabela pode ter “n” relacionamentos com outra tabela - um pai pode ter “n’ filhos)

N:1 (idem anterior).

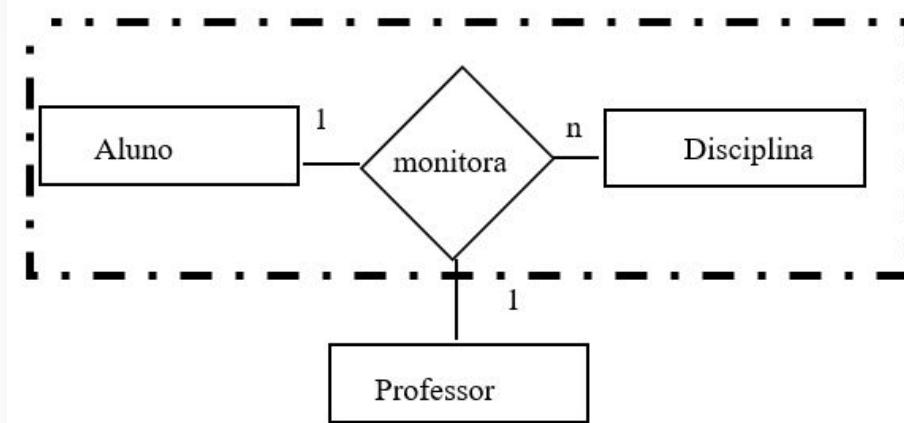
N:N ou N:M (muitos para muitos) - 1 aluno cursa “n” disciplinas e uma disciplina pode conter “n” alunos).

Relacionamentos

Grau do relacionamento (número de entidades no relacionamento)

2 entidades => binário

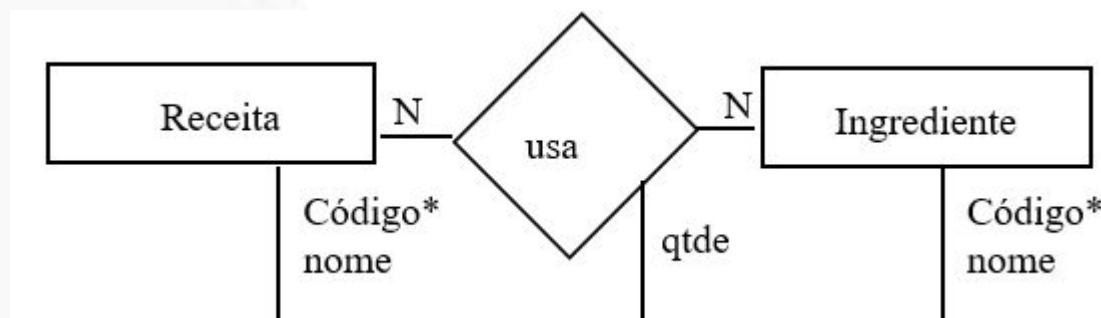
3 entidades => ternário



Relacionamento N:N

N:N (muitos para muitos) - 1 receita usa “n” ingredientes e um ingrediente pode estar em “n” receitas.

Modelo Lógico com duas entidades. Modelo Físico passa a ter a entidade Receita_Ingrediente.



Relacionamento N:N



N:N (muitos para muitos) - 1 nota fiscal de venda tem “n” produtos e um produto pode estar em “n” notas fiscais de venda.

Modelo Peter Chen



1. Identificar
 - Entidades.
 - Relacionamentos.
 - Atributos.
2. Desenhar Modelo E-R.
3. Mapear DER para o modelo relacional.
4. Definir estrutura dos registros.

Estrutura dos Registros

Detalhar cada arquivo e seus campos.

- **Tipo** (caractere/numérico/data/moeda).
- **Máscara** (CPF, CNPJ).
- **Valor padrão ou “default”** (data atual).
- **Tamanho** (30 , 40 ou 50 espaços para o nome).
- **Regra de validação**: faixa de valores aceitáveis ou domínio
(tamanho: P, M, G).

Conclusão



- ✓ Normalização e desnormalização
- ✓ Entidades
- ✓ Atributos
- ✓ Relacionamentos

Próxima Aula



01. ••

SQL

02. ••

03. ••

04. ••

Armazenamento de Dados

Capítulo 01 – Aula 01.05 – SQL

PROF.: RICARDO BRITO ALVES

Nesta Aula



- DDL
- DML
- DQL
- DTL
- DCL

SQL - Structured Query Language



O Modelo Relacional prevê, desde sua concepção, *a existência de uma linguagem baseada em caracteres que suporte a definição do esquema físico (tabelas, restrições, etc.), e sua manipulação (inserção, consulta, atualização e remoção).*

Linguagem SQL



DDL - Data Definition Language - Linguagem de Definição de Dados.

São os comandos que *interagem com os objetos do banco*.

São comandos DDL : CREATE, ALTER e DROP

Linguagem SQL



DDL - Data Definition Language - Linguagem de Definição de Dados.

```
CREATE TABLE IF NOT EXISTS contatos (
    nome VARCHAR(20) NOT NULL,
    sobrenome VARCHAR(30) NOT NULL,
    ddd INT(2) NOT NULL,
    telefone VARCHAR(9) NOT NULL,
    data_nasc DATE NULL,
    email VARCHAR(30) NULL);
```

Linguagem SQL



DDL - Data Definition Language - Linguagem de Definição de Dados.

ALTER TABLE contatos

ADD ativo SMALLINT NOT NULL AFTER email;

ALTER TABLE contatos

CHANGE telefone fone VARCHAR(9) NOT NULL;

Linguagem SQL



DDL - Data Definition Language - Linguagem de Definição de Dados.

DROP TABLE contatos

Linguagem SQL



DML - Data Manipulation Language - Linguagem de Manipulação
de Dados.

São os comandos que *interagem com os dados dentro das tabelas*.

São comandos DML : INSERT, UPDATE e DELETE

Linguagem SQL



DML - Data Manipulation Language - Linguagem de Manipulação
de Dados.

```
INSERT INTO contatos (nome,sobrenome,ddd,telefone,data_nasc  
,email,ativo)  
  
VALUES('Bruno','Santos',11,999999999,'2015-08-22'  
, 'contato@dominio.com.br',1);
```

Linguagem SQL



DML - Data Manipulation Language - Linguagem de Manipulação
de Dados.

UPDATE contatos SET

sobrenome= 'Nascimento'

, ddd= 015

, telefone= '0123456789'

WHERE nro_contato = 100

Linguagem SQL



DML - Data Manipulation Language - Linguagem de Manipulação
de Dados.

DELETE

Linguagem SQL



DQL - Data Query Language - Linguagem de Consulta de dados.

São os comandos de consulta.

São comandos DQL : SELECT (é o comando de consulta)

Alguns autores consideram que o SELECT fica na DML.

Recentemente há a visão que o SELECT faz parte da DQL.

Linguagem SQL



DQL - Data Query Language - Linguagem de Consulta de dados.

```
SELECT nome, sobrenome
```

```
FROM contatos
```

```
WHERE nro_contato= 100;
```

> maior

< menor

>= maior e igual

<= menor e igual

<> diferente

Linguagem SQL



DTL - Data Transaction Language - Linguagem de Transação de Dados.

São os comandos ***para controle de transação***.

São comandos DTL : BEGIN TRANSACTION, COMMIT E ROLLBACK.

```
UPDATE contatos SET sobrenome= 'Nascimento'
```

```
WHERE nro_contato= 100;
```

```
commit;
```

```
rollback;
```

Linguagem SQL



DCL - Data Control Language - Linguagem de Controle de Dados.

São os comandos *para controlar a parte de segurança do banco de dados.*

São comandos DCL : GRANT, REVOKE.

Você pode conceder privilégios GRANT e revogar REVOKE em vários objetos de banco de dados no MySQL. Você pode então ver os privilégios atribuídos a um usuário usando o comando SHOW GRANTS.

Linguagem SQL



DCL - Data Control Language

GRANT privileges ON object TO user;

GRANT EXECUTE ON [PROCEDURE | FUNCTION] object TO user;

GRANT SELECT, INSERT, UPDATE, DELETE ON contacts TO 'smithj'@'localhost';

GRANT ALL ON contacts TO 'smithj'@'localhost';

GRANT SELECT ON contacts TO '*'@'localhost';

Privilege	Description
SELECT	Ability to perform SELECT statements on the table.
INSERT	Ability to perform INSERT statements on the table.
UPDATE	Ability to perform UPDATE statements on the table.
DELETE	Ability to perform DELETE statements on the table.
INDEX	Ability to create an index on an existing table.
CREATE	Ability to perform CREATE TABLE statements.
ALTER	Ability to perform ALTER TABLE statements to change the table definition.
DROP	Ability to perform DROP TABLE statements.
GRANT OPTION	Allows you to grant the privileges that you possess to other users.
ALL	Grants all permissions except GRANT OPTION.

Linguagem SQL



DCL - Data Control Language

REVOKE privileges ON object FROM user;

REVOKE EXECUTE ON [PROCEDURE | FUNCTION] object FROM user;

REVOKE DELETE, UPDATE ON contacts FROM 'smithj'@'localhost';

REVOKE ALL ON contacts FROM 'smithj'@'localhost';

REVOKE SELECT ON contacts FROM '*'@'localhost';

Conclusão



Linguagem SQL e suas divisões:

- ✓ DDL
- ✓ DML
- ✓ DQL
- ✓ DTL
- ✓ DCL

Próxima Aula



01. ••

Instalando o BR Modelo

02. ••

03. ••

04. ••

Armazenamento de Dados

Capítulo 01 – Aula 01.06 – Instalando o BR Modelo

PROF.: RICARDO BRITO ALVES

Nesta Aula



- Instalação BR Modelo

BR Modelo



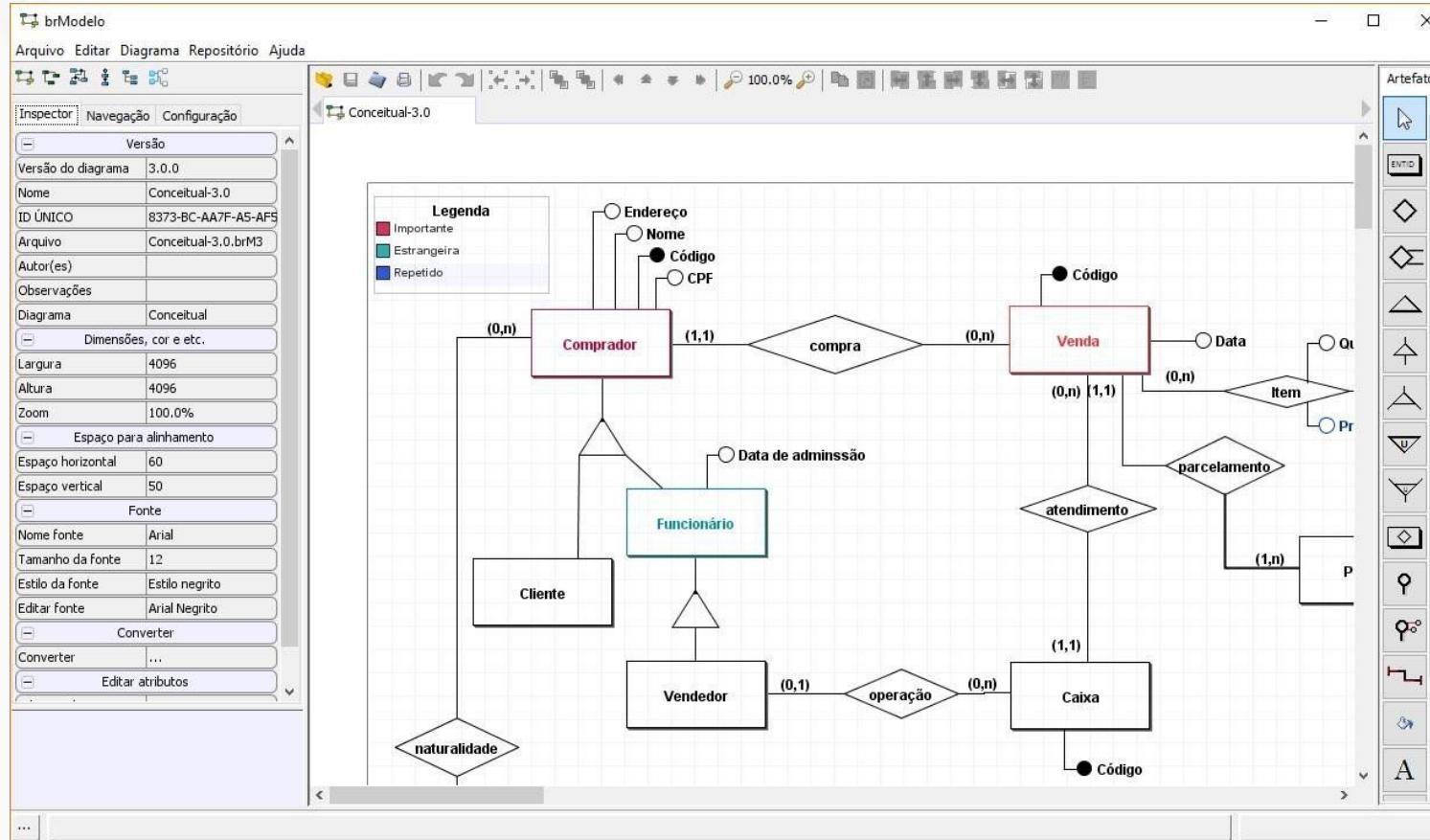
Ferramenta para modelagem ER em bancos de dados.

Características

- Modelo ER.
- SQL e DDL.
- Modelo de banco de dados físico.
- Modelo lógico de banco de dados.
- Modelo de banco de dados conceitual.

BR Modelo

IGTI



BR Modelo



A instalação é bem simples.

Basta baixar uma versão do site:

<https://sourceforge.net/projects/brmodelo/>

Você terá um arquivo chamado brModelo.jar. Basta clicar duas vezes nesse arquivo **jar** para executar o BR Modelo.

Conclusão



Instalação BR Modelo

Próxima Aula



01.

03.

Instalação do Ambiente de Estudo

02.

04.

Armazenamento de Dados

Capítulo 01 – Aula 01.07 – Instalando o Ambiente de Estudo

PROF.: RICARDO BRITO ALVES

Nesta aula



- Instalação
 - MySQL Workbench
 - MySQL XAMPP
 - Pentaho

Conclusão



Instalação:

- ✓ MySQL Workbench
- ✓ MySQL XAMPP
- ✓ Pentaho

Próxima aula



01. ••

Prática com MySQL

02. ••

03. ••

04. ••

Armazenamento de Dados

Capítulo 01 – Aula 01.08 – Prática com MySQL

PROF.: RICARDO BRITO ALVES

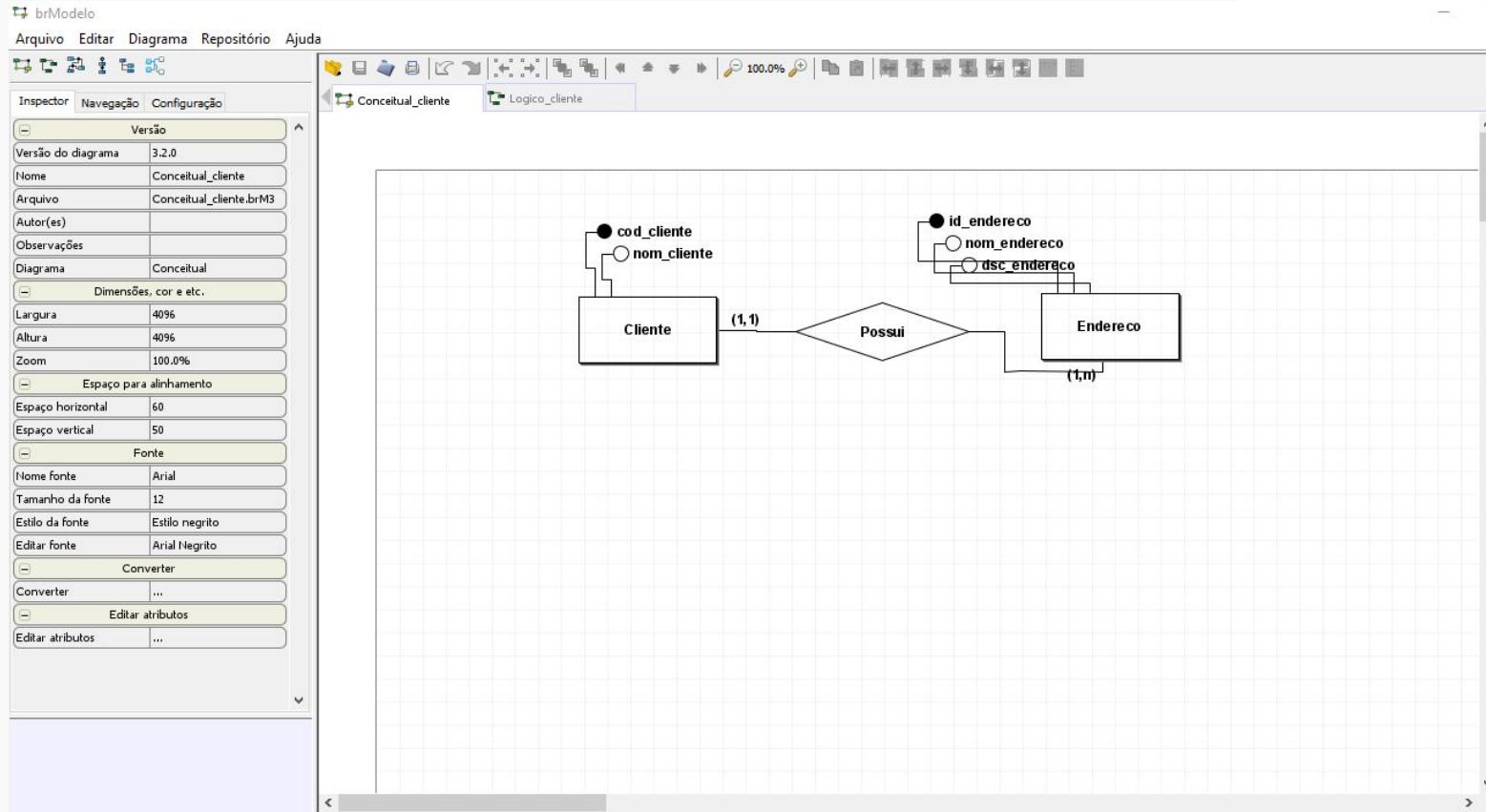
Nesta aula



- Prática com MySQL

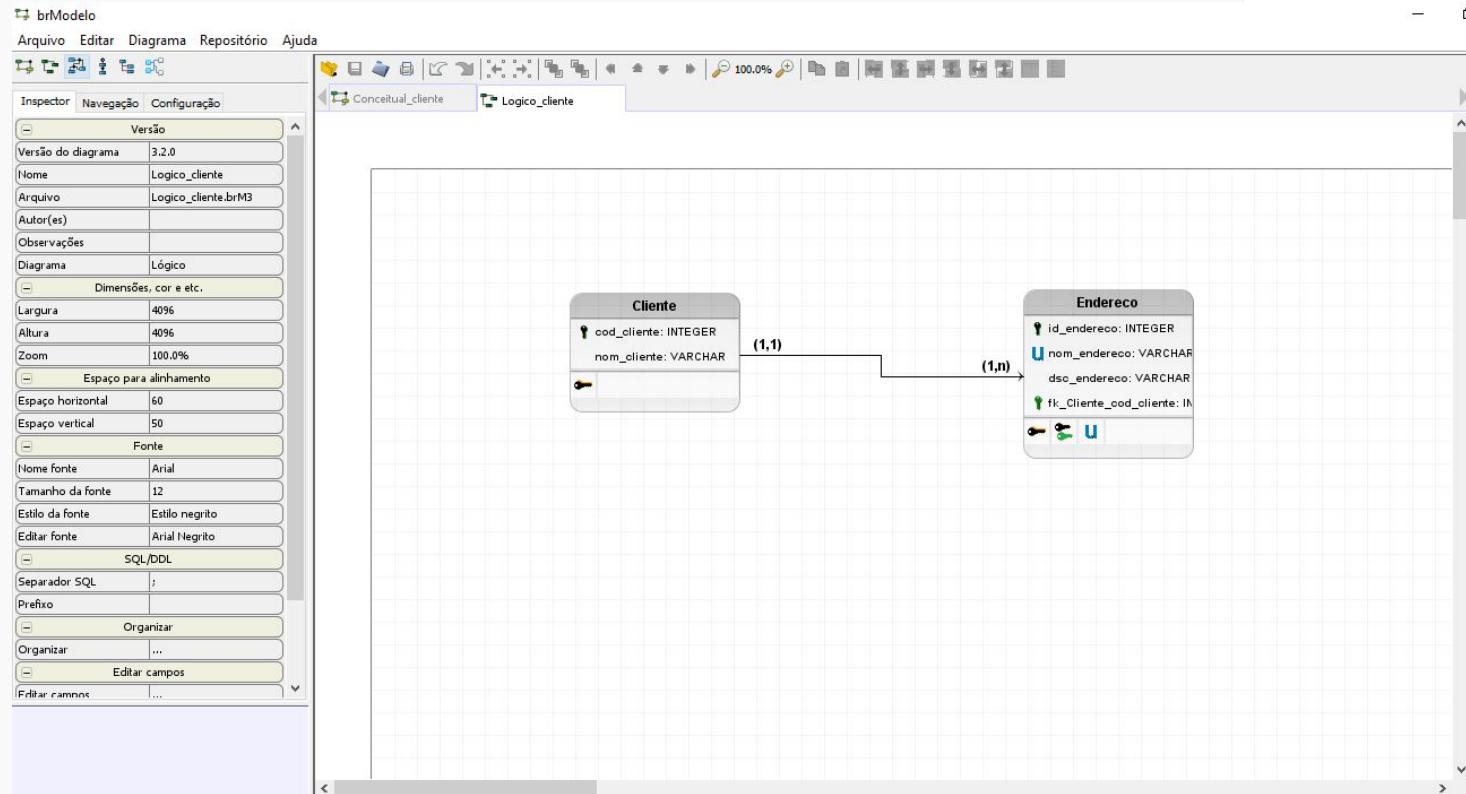
BR Modelo

IGTI



BR Modelo

IGTI



BR Modelo

IGTI

Exibição de modelo físico

1. /* Logico_cliente: */
2.
3. CREAT TABLE Cliente (
4. cod_cliente INTEGER PRIMARY KEY,
5. nom_cliente VARCHAR
6.);
7.
8. CREAT TABLE Endereco (
9. id_endereco INTEGER PRIMARY KEY,
10. nom_endereco VARCHAR UNIQUE,
11. dsc_endereco VARCHAR,
12. fk_Cliente_cod_cliente INTEGER
13.);
14.
15. ALTE TABLE Endereco ADD CONSTRAINT FK_Endereco_2
16. FOREIGN KEY (fk_Cliente_cod_cliente)
17. REFERENCES Cliente (cod_cliente)
18. ON DELETE RESTRICT;

Fechar

MySQL Workbench



MySQL Workbench

Local instance MySQL80 ×

File Edit View Query Database Server Tools Scripting Help

Navigator: teste - Schema

SCHEMAS

Filter objects

- cidade
- cliente
- curso
- disciplina
- disciplina_has_aluno
- endereco
- estado
- Views
- Stored Procedures
- Functions
- mydb
- stage
- stage1
- sys
- teste
 - Tables
 - cliente
 - endereco
 - Views

Administration Schemas

Information

No object selected

Query 1

```
1 /* Logico_cliente: */
2
3 • CREATE TABLE teste.Cliente (
4     cod_cliente INT PRIMARY KEY,
5     nom_cliente VARCHAR(30)
6 );
7
8 • CREATE TABLE teste.Endereco (
9     id_endereco INT PRIMARY KEY,
10    nom_endereco VARCHAR(10) UNIQUE,
11    dsc_endereco VARCHAR(60),
12    fk_Cliente_cod_cliente INTEGER
13 );
14
15 • ALTER TABLE teste.Endereco ADD CONSTRAINT FK_Endereco_2
16     FOREIGN KEY (fk_Cliente_cod_cliente)
17     REFERENCES teste.Cliente (cod_cliente)
18     ON DELETE RESTRICT;
```

Conclusão



Aula prática com

- ✓ BR Modelo
- ✓ MySQL

Próxima aula



01. ••

Banco de Dados NoSQL

02. ••

03. ••

04. ••

Armazenamento de Dados

Capítulo 02 – Banco de Dados NoSQL

PROF.: RICARDO BRITO ALVES

Armazenamento de Dados

Capítulo 02 – Aula 02.01 – Visão geral de NoSQL

PROF.: RICARDO BRITO ALVES

Nesta Aula



- ❑ Definição de NoSQL
- ❑ Características
- ❑ Alguns bancos NoSQL

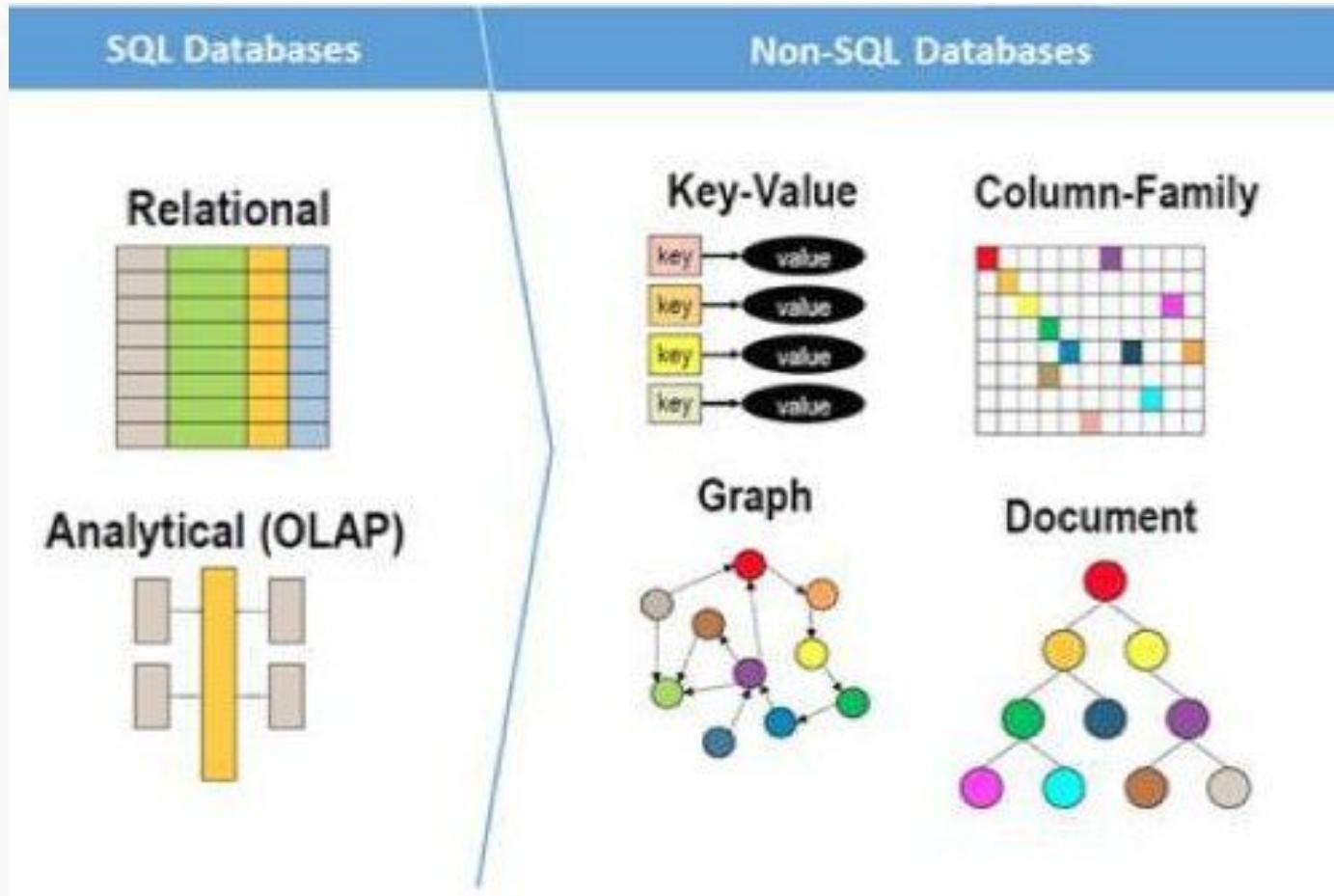
Banco de Dados Relacional



- Os dados são estruturados de acordo com o modelo relacional.
- SQL Server, Oracle, PostgreSQL, MySQL, DB2 e outros.
- Elementos básicos
 - Relações (tabelas) e registros (tuplas)
- Características fundamentais
 - ✓ Restrições de integridade
 - ✓ PK-primary key, FK-foreign key, UK-unique key, CK-check key, NN-not null
 - ✓ Normalização (formas normais)
 - ✓ Linguagem SQL (Structured Query Language)

Banco de Dados Relacional

IGTI



Ranking Banco de Dados

<https://db-engines.com/en/ranking>

Rank Aug 2020	Jul 2020	Aug 2019	DBMS	Database Model	Score		
					Aug 2020	Jul 2020	Aug 2019
1.	1.	1.	Oracle +	Relational, Multi-model 	1355.16	+14.90	+15.68
2.	2.	2.	MySQL +	Relational, Multi-model 	1261.57	-6.93	+7.89
3.	3.	3.	Microsoft SQL Server +	Relational, Multi-model 	1075.87	+16.15	-17.30
4.	4.	4.	PostgreSQL +	Relational, Multi-model 	536.77	+9.76	+55.43
5.	5.	5.	MongoDB +	Document, Multi-model 	443.56	+0.08	+38.99
6.	6.	6.	IBM Db2 +	Relational, Multi-model 	162.45	-0.72	-10.50
7.	↑ 8.	↑ 8.	Redis +	Key-value, Multi-model 	152.87	+2.83	+8.79
8.	↓ 7.	↓ 7.	Elasticsearch +	Search engine, Multi-model 	152.32	+0.73	+3.23
9.	9.	↑ 11.	SQLite +	Relational	126.82	-0.64	+4.10
10.	↑ 11.	↓ 9.	Microsoft Access	Relational	119.86	+3.32	-15.47
11.	↓ 10.	↓ 10.	Cassandra +	Wide column	119.84	-1.25	-5.37
12.	12.	↑ 13.	MariaDB +	Relational, Multi-model 	90.92	-0.21	+5.96
13.	13.	↓ 12.	Splunk	Search engine	89.91	+1.64	+4.03
14.	↑ 15.	↑ 15.	Teradata +	Relational, Multi-model 	76.78	+0.81	+0.14
15.	↓ 14.	↓ 14.	Hive	Relational	75.29	-1.14	-6.51

NoSQL

IGTI

NoSQL é um termo genérico que define **bancos de dados não-relacionais**.

A tecnologia NoSQL foi iniciada por companhias líderes da Internet como Google, Facebook, Amazon e LinkedIn, para superar as limitações de banco de dados relacional para aplicações web modernas.

SQL

Table: Users			
	id	name	createdAt
	Filter	Filter	Filter
1	1	Ashton	2018-08-12 1...
2	2	Jakeem	2018-08-12 1...
3	3	Uma	2018-08-12 1...

NoSQL



Users Collection



NoSQL

Não quer dizer que seus modelos não possuem relacionamentos e sim que *não são orientados a tabelas*.

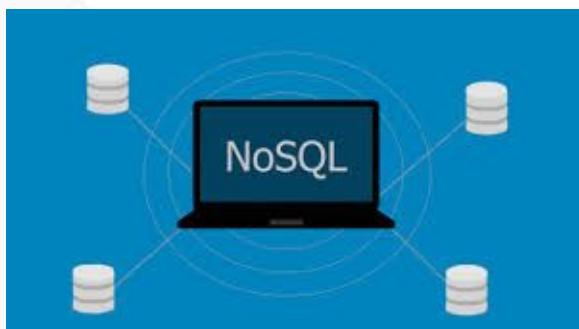
Not Only SQL (não apenas SQL).

Bancos de dados NoSQL são cada vez mais usados em *Big Data* e *aplicações web de tempo real*.



NoSQL - Características

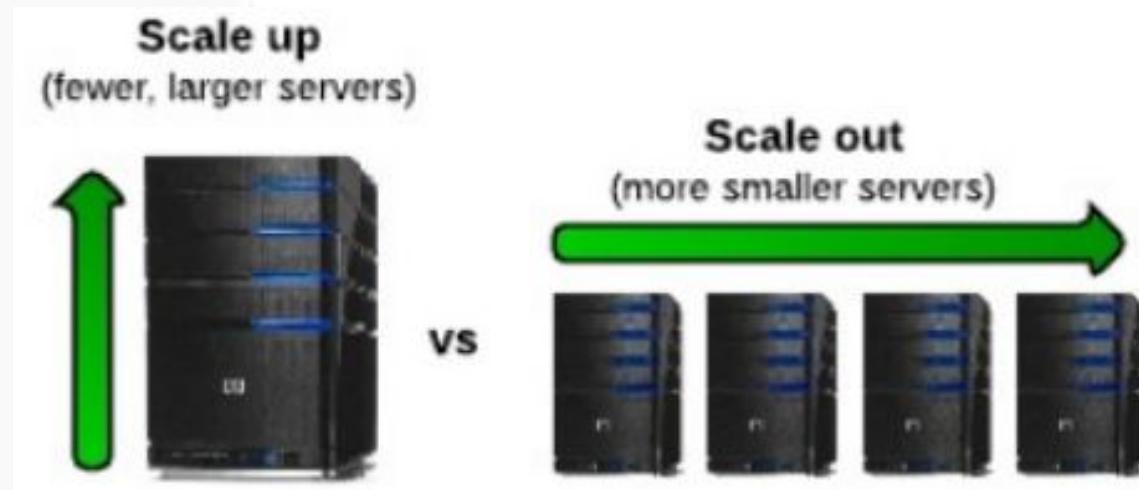
- Escalabilidade horizontal.
- Ausência de esquema ou esquema flexível.
- Suporte a replicação.
- API simples.
- Nem sempre prima pela consistência.



Escalabilidade Horizontal

À medida em que o volume de dados cresce, aumenta-se a necessidade de escalabilidade e melhoria do desempenho.

Podemos escalar **verticalmente** (adicionar CPU, memória, disco) ou podemos escalar **horizontalmente** (adicionar mais nós).

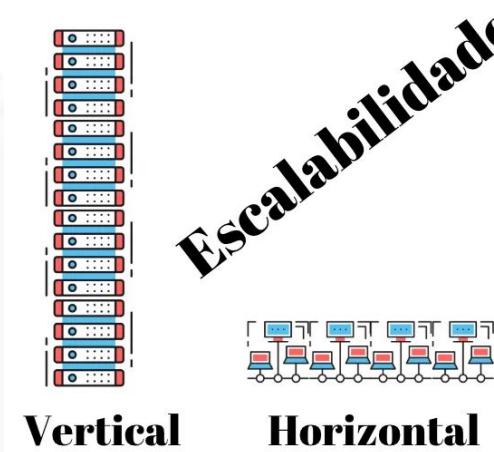


Escalabilidade Horizontal

Dentre todas as possibilidades para esta solução, **a escalabilidade horizontal se torna a mais viável, porém requer diversas threads ou que processos de uma tarefa sejam criadas e distribuídas.**

Não é que os bancos de dados relacionais não escalam, eles não escalam facilmente.

Como nos modelos NoSQL não existe bloqueios, esse tipo de escalabilidade se torna mais viável.



Escalabilidade Horizontal

Uma alternativa muito utilizada para alcançar a escalabilidade horizontal é o **Sharding**, que *divide os dados em múltiplas tabelas a serem armazenadas ao longo de diversos nós na rede*.

Um shard de banco de dados, ou literalmente um fragmento de banco de dados, é uma partição horizontal de dados em um banco de dados ou mecanismo de busca.

Cada partição individual é referenciada como um shard ou shard de banco de dados.

Sharding

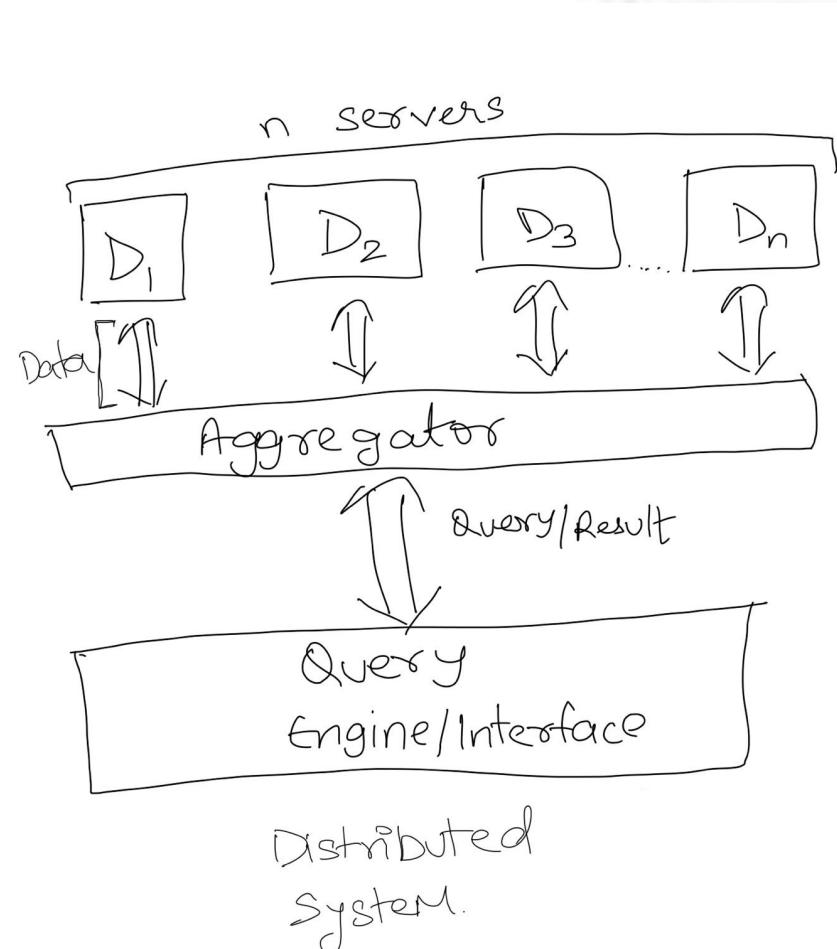
Como o **Sharding** ajuda a obter

Você pode particionar um índice em um servidor separado.

Se você consultar um servidor, obterá um conjunto completo de dados. **O sharding distribuída usa um agregador** que combina. **Um agregador por servidor.**

Esse programa agregador é o que faz com que o sistema seja distribuído.

Em outras palavras, **Sistemas**



Ausência de Esquema

Ausência de esquema (Schema-free) ou esquema flexível é uma outra característica em bancos de dados NoSQL.

Basicamente é a *ausência parcial ou total de esquema que define a estrutura de dados*.

É justamente essa ausência de esquema que *facilita uma alta escalabilidade* e alta disponibilidade, *mas em contrapartida não há a garantia de integridade dos dados, fato este que não ocorre no Modelo Relacional*.

Ausência de Esquema

ID	Name	IsActive	Dob
1	John Smith	True	8/30/1964
2	Sarah Jones	False	2/18/2002
3	Adam Stark	True	7/13/1987

Document 1

```
{  
  "id": "1",  
  "name": "John Smith",  
  "isActive": true,  
  "dob": "1964-30-08"  
}
```

Document 2

```
{  
  "id": "2",  
  "fullName": "Sarah Jones",  
  "isActive": false,  
  "dob": "2002-02-18"  
}
```

Document 3

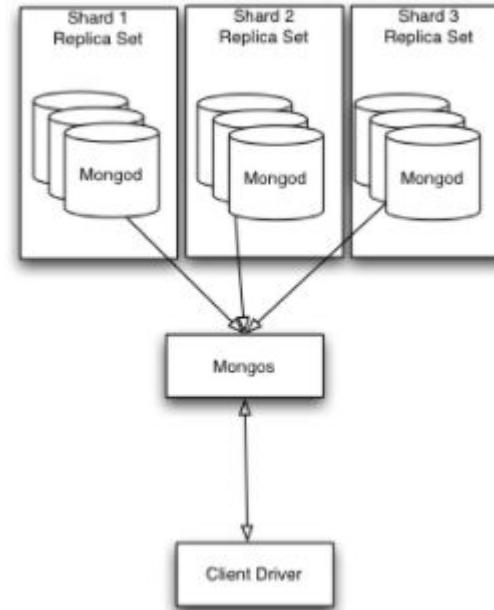
```
{  
  "id": "3",  
  "fullName":  
  {  
    "first": "Adam",  
    "last": "Stark"  
  },  
  "isActive": true,  
  "dob": "2015-04-19"  
}
```

Replicação

Supor o nativo a replic o o  o o outra forma de prover a escalabilidade, pois, no momento em que permitimos a replic o o de forma nativa o tempo gasto para recuperar informa o es  o o reduzido.

Replicação refere-se ao armazenamento de dados e a estratégia de backups entre computadores em locais distintos.

É um conjunto de tecnologias utilizadas ***para copiar e distribuir objetos e dados de um banco de dados para outro banco de dados***, sincronizando estes dados com a ***finalidade de se manter a consistência***.



Replicação

No caso dos bancos de dados SQL que possuem as propriedades ACID (Atomicidade, Consistência, Isolamento e Durabilidade) o modelo normalmente utilizado é o ***master-slave (mestre-escravo)*** em que um servidor atua como master e os demais atuam como slave onde todas as operações que alteram algum dado no master são repassadas de imediato para os servidores slaves.

Um exemplo de banco de dados que utiliza este tipo de replicação é o MySQL.

Replicação

No caso dos bancos de dados NoSQL que trabalham com a propriedade BASE (Basicamente Disponível Eventualmente Consistente) o modelo normalmente adotado é o ***multimaster*** onde as operações de leitura e gravação podem ser realizadas através de qualquer nó.

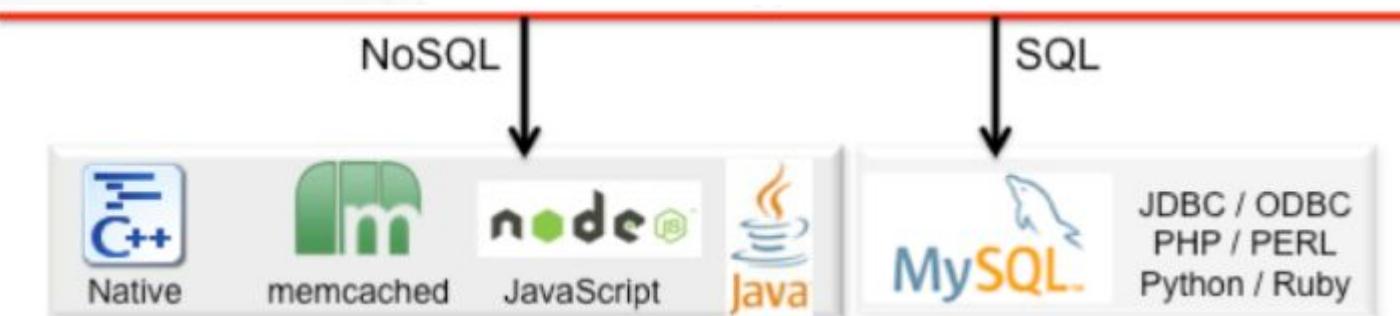
Esse tipo de banco de dados possui fraca consistência e tem como fator principal a disponibilidade.

Um exemplo de banco de dados que trabalha com este tipo de replicação é o ***Cassandra*** que ***usa o particionamento dos dados através da estratégia de replicação e da topologia dos clusters (distribuição dos nós).***

API

API simples para acessar o banco de dados, ou seja, em banco de dados NoSQL, o foco não está no armazenamento dos dados e sim como recuperar estes dados de forma eficiente.

Pensando nisso, é fundamental ter APIs desenvolvidas para facilitar o acesso às informações de forma rápida e eficiente.

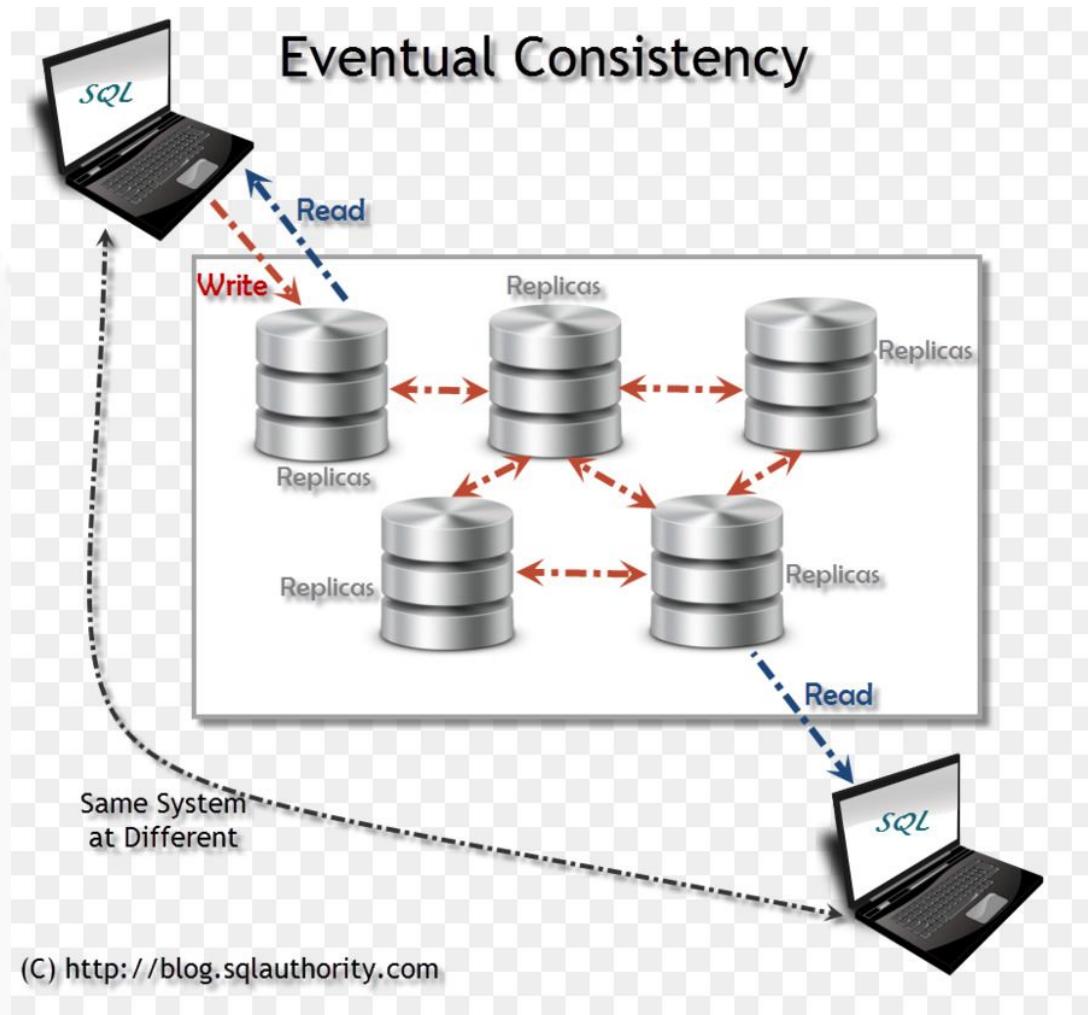


Consistência

Consistência eventual é outra característica particular de bancos NoSQL onde nem sempre a consistência dos dados é mantida.

É necessário haver um planejamento para que o sistema possa tolerar inconsistências temporárias com o objetivo de priorizar a disponibilidade.

Consistência



Alguns Bancos NoSQL



Aerospike: Banco de dados NoSQL que oferece *uma vantagem de velocidade de memória, atraindo empresas de anúncios de alta escala e aquelas que precisam de tempos de resposta em milissegundo.* Aerospike está apostando em novas categorias, incluindo jogos, e-commerce e segurança, onde a baixa latência é tudo.

Apache Cassandra: Os pontos fortes são a modelagem de dados NoSQL e *escalabilidade linear flexível em hardware* por conta do uso de cluster.

Amazon DynamoDB: foi desenvolvido pela Amazon para incrementar o seu e-commerce, possibilitando ter seus serviços altamente escaláveis. Inspirou o Cassandra, Riak, e outros projetos NoSQL.

Alguns Bancos NoSQL



MongoDB: É o banco de dados mais popular NoSQL, com mais de sete milhões de downloads e centenas de milhares de implantações. ***Sua popularidade se deve à facilidade de desenvolvimento e manejo flexível dos dados.*** Muito utilizado em aplicações de redes sociais web e móvel.

HBase: É o banco de dados que roda em cima do ***HDFS (Hadoop Distributed File System – sistema de arquivos distribuído)***, por isso dá aos usuários a capacidade única de trabalhar diretamente com os dados armazenados no Hadoop. As características incluem grande escalabilidade.

Redis: É o banco de dados NoSQL do tipo chave-valor mais conhecido. No mercado podemos encontrar diversas outras soluções que também são mecanismos de armazenamento baseado em chave-valor.

Conclusão



- ✓ Definição de NoSQL
- ✓ Características do NoSQL
- ✓ Alguns bancos NoSQL

Próxima Aula



01. •

Propriedades de Bancos de
Dados

02. •

03. •

04. •

Armazenamento de Dados

Capítulo 02 – Aula 02.02 – Propriedades dos Banco de Dados

PROF.: RICARDO BRITO ALVES

Nesta Aula



- Propriedades
 - ACID x Base
 - Teorema CAP

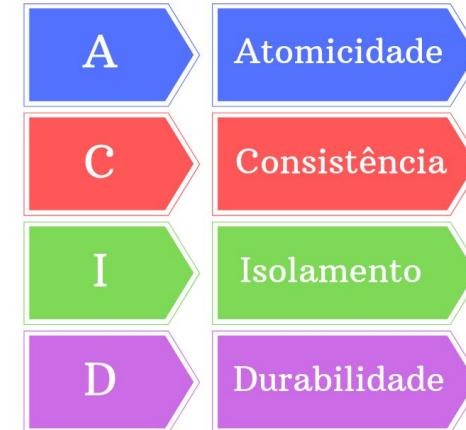
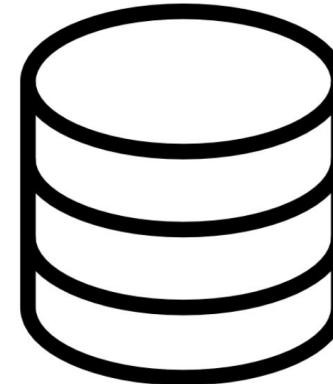
Propriedades ACID

Atomicidade - estado em que as modificações no BD devem ser todas ou nenhuma feita. Cada transação é dita como “atômica”. Se uma parte desta transação falhar, toda transação falhará.

Consistência - estado que garante que todos os dados serão escritos no BD.

Isolamento - requer que múltiplas transações que estejam ocorrendo “ao mesmo tempo”, não interfiram nas outras.

Durabilidade - garante que toda transação submetida (commit) pelo BD não será perdida



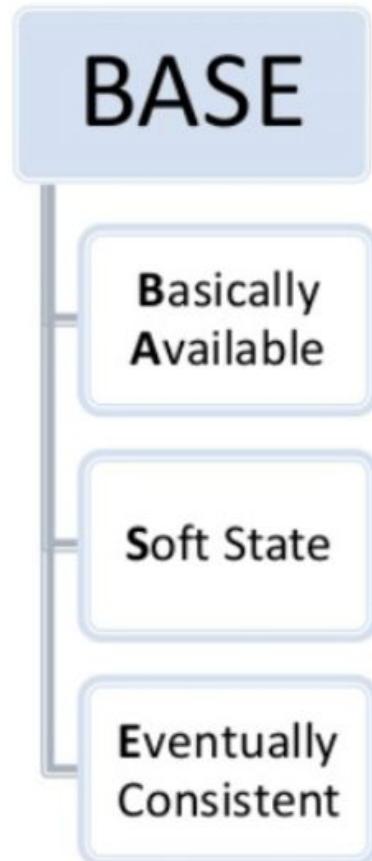
Propriedades BASE

Basically Available – Basicamente Disponível.

Soft-State – Estado Leve

Eventually Consistent – Eventualmente Consistente.

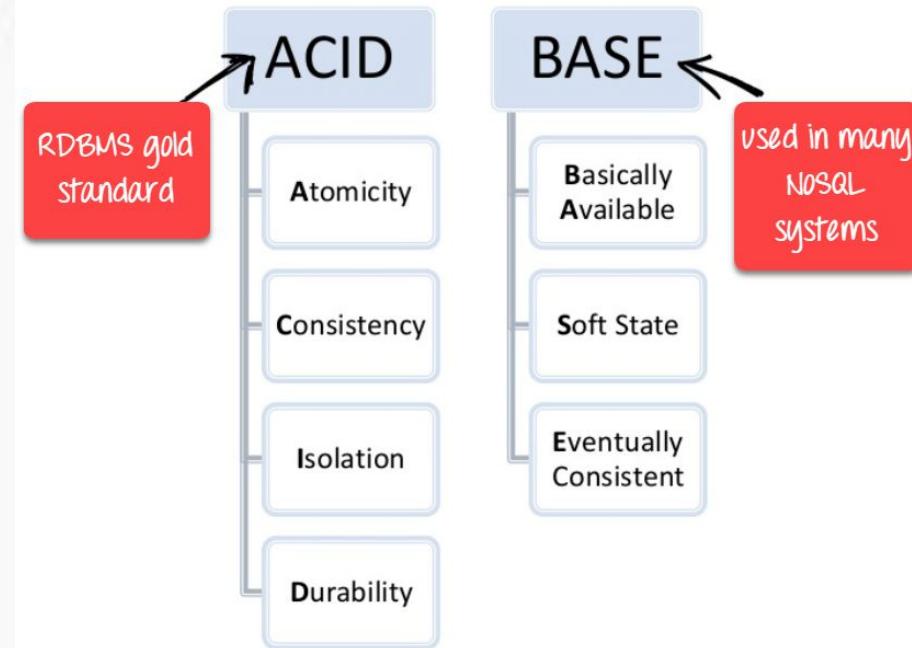
Uma aplicação funciona basicamente todo o tempo (Basicamente Disponível), não tem de ser consistente todo o tempo (Estado Leve) e o sistema torna-se consistente no momento devido (Eventualmente Consistente).



ACID vs BASE

BDs Relacionais trabalham com **ACID** (Atomicidade, Consistência, Isolabilidade, Durabilidade).

BDs NoSQL trabalham com **BASE** (Basicamente Disponível, Estado Leve, Eventualmente consistente).



CAP



Consistency – Consistência

Availability – Disponibilidade

Partition Tolerance – Tolerância ao Particionamento

Consistência

Consistência (**Consistent**)

As escritas são atômicas e todas as requisições de dados subsequentes obtém o novo valor. Assim, é garantido o retorno do dado mais atualizado armazenado, logo após ter sido escrito. Isso independe de qual nó seja consultado pelo cliente – o dado retornado para a aplicação será igual.

Disponibilidade

Disponibilidade (**Available**)

O banco de dados sempre retornará um valor desde que ao menos um servidor esteja em execução – mesmo que seja um valor defasado.

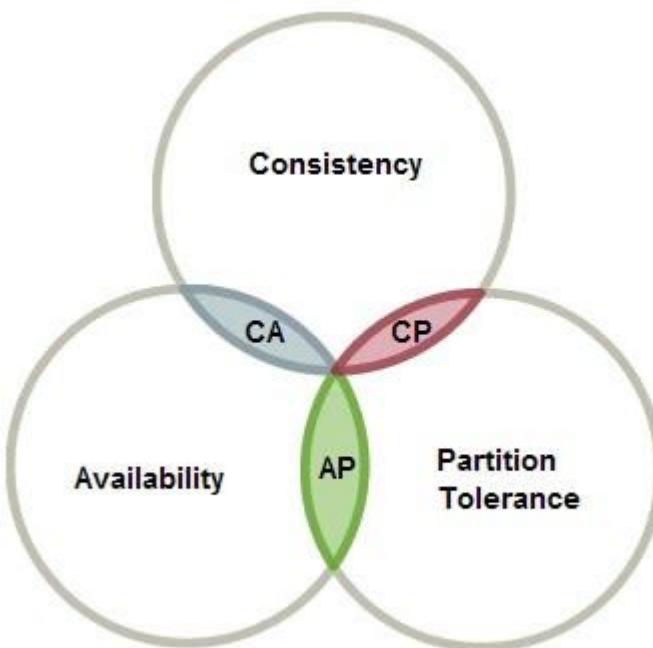
Tolerância ao Particionamento

Tolerância ao Particionamento (**Partition Tolerant**) ou Tolerante a Falhas.

O sistema ainda irá funcionar mesmo se a comunicação com o servidor for temporariamente perdida. Assim, se houver falha de comunicação com um nó da rede distribuída, os outros nós poderão responder às solicitações de consultas de dados dos clientes.

Teorema CAP

De acordo com o teorema **CAP**, *um sistema distribuído de bancos de dados somente pode operar com dois desses comportamentos ao mesmo tempo, mas jamais com os três simultaneamente.*

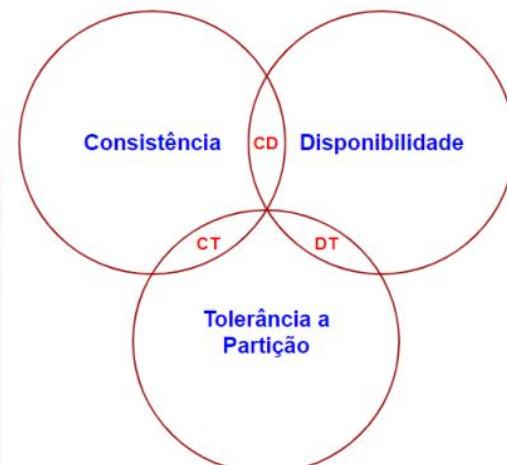


Consistência Eventual

É um conceito interessante derivado do teorema CAP.

O sistema prioriza as escritas de dados (armazenamento), sendo o sincronismo entre os nós do servidor realizado em um momento posterior – o que causa um pequeno intervalo de tempo no qual o sistema como um todo é inconsistente.

Para isso, são implementadas as propriedades Disponibilidade e Tolerância a Partição.



Consistência Eventual

Exemplos de sistemas de bancos de dados que implementam a consistência eventual são o ***MongoDB, Cassandra e RavenDB (bancos NoSQL)***, entre outros.

Em Bancos Relacionais, é muito comum implementar as propriedades **Consistência** e **Disponibilidade**. Como exemplos, citamos os SGBDRs ***Oracle, MySQL, PostgreSQL, SQL Server*** e outros.

Ao criar um banco de dados distribuído é importante ***ter em mente o teorema CAP***.

Você terá de decidir se o banco será consistente ou disponível, pois bancos de dados distribuídos são sempre tolerantes a partição.

Conclusão



Propriedades dos bancos de dados

- ✓ ACID
- ✓ BASE
- ✓ Teorema CAP

Próxima Aula



01. ••

NewSQL

02. ••

03. ••

04. ••

Armazenamento de Dados

Capítulo 02 – Aula 02.03 – Visão geral de NewSQL

PROF.: RICARDO BRITO ALVES

Nesta Aula



- ❑ Definição e visão de NewSQL
- ❑ Exemplo de bancos de dados NewSQL

NewSQL

Pode ser definido como uma classe de SGBDs relacionais modernos que buscam fornecer o mesmo desempenho escalonável do NoSQL para cargas de trabalho OLTP e, simultaneamente, garantir a conformidade ACID para transações como no RDBMS.

Basicamente esses sistemas desejam alcançar a escalabilidade do NoSQL sem ter que descartar o modelo relacional com SQL e suporte a transações do DBMS legado.



Conceitos do NewSQL



- *Expansão dividindo um banco de dados em subconjuntos separados chamados partições ou fragmentos*, levando à execução de uma consulta em várias partições e, em seguida, combinar o resultado em um único resultado.
- Os novos sistemas SQL *preservam as propriedades ACID* dos bancos de dados.
- Benefícios do sistema de *controle de simultaneidade aprimorado* em relação aos tradicionais.

Conceitos do NewSQL



- *Presença de índice secundário permitindo NewSQL para suportar tempos de processamento de consulta mais rápidos.*
- Alta disponibilidade e durabilidade de dados forte são possíveis com o uso de *mecanismos de replicação*.
- Novos sistemas SQL podem ser configurados para fornecer *atualizações síncronas de dados* pela WAN.
- *Minimiza o tempo de inatividade, fornece tolerância a falhas com seu mecanismo de recuperação de falha.*

SQL, NoSQL ou NewSQL



- O SQL está em conformidade com as propriedades ACID e funciona bem com ***escalabilidade vertical***.
- O NoSQL oferece seu próprio ***escalonamento horizontal*** e está em conformidade com propriedades BASE. NoSQL, no entanto, não obedece às regras ACID, que são necessárias para manter um banco de dados confiável e consistente.
- As empresas e organizações em ritmo acelerado geram terabytes de dados transacionais diariamente enquanto trabalham em um sistema OLTP. NewSQL é a escolha ideal. ***NewSQL melhora o SQL ao fornecer escalabilidade horizontal enquanto mantém as propriedades ACID***.

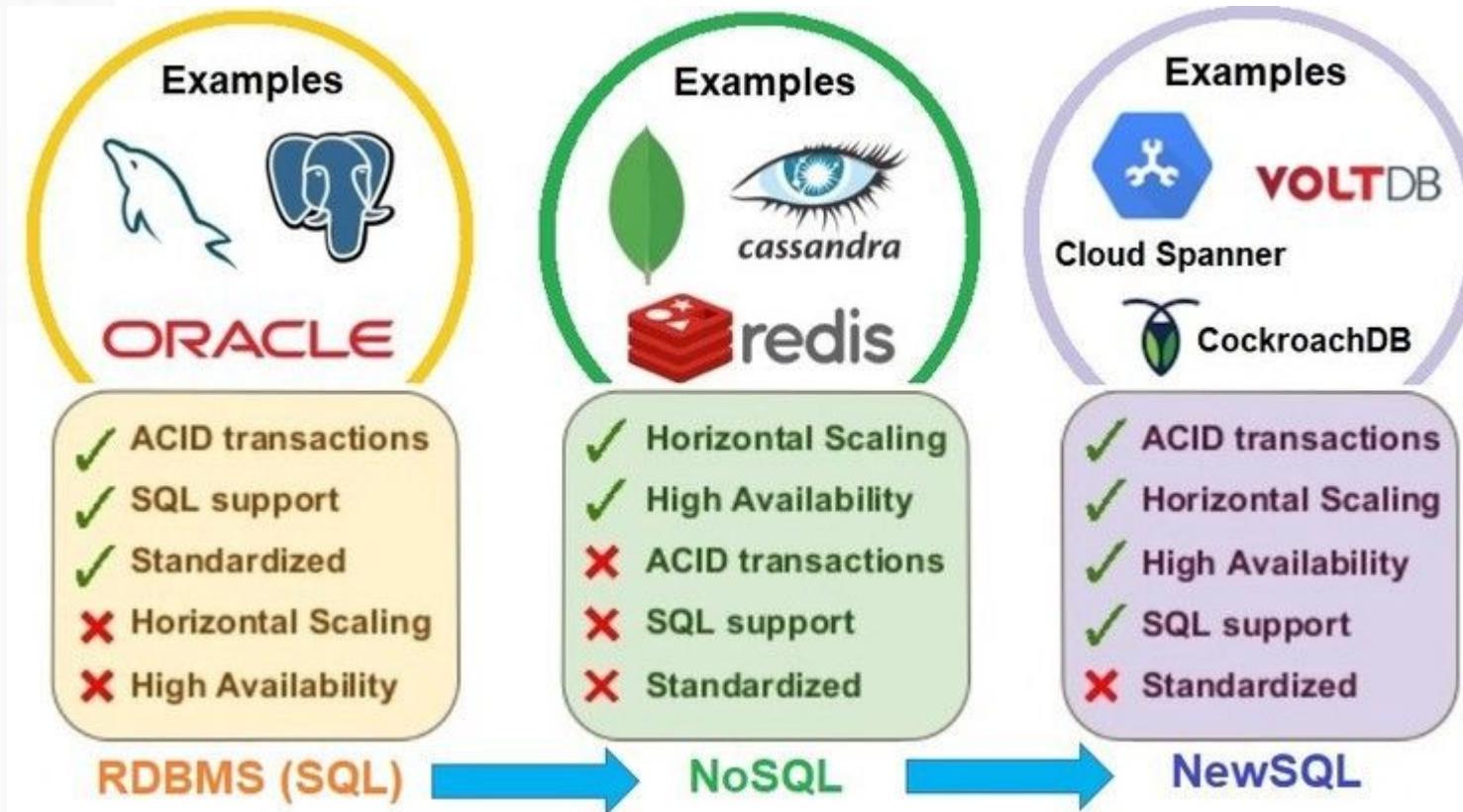
SQL, NoSQL ou NewSQL

IGTI

	Old SQL	NoSQL	NewSQL
Relational	Yes	No	Yes
SQL	Yes	No	Yes
ACID transactions	Yes	No	Yes
Horizontal scalability	No	Yes	Yes
Performance / big volume	No	Yes	Yes
Schema-less	No	Yes	No

SQL, NoSQL ou NewSQL

IGTI



Alguns Bancos NewSQL



MemSQL: Como o próprio nome sugere, é *operado em memória*, e é um sistema de banco de dados de alta escala por sua combinação de desempenho e compatibilidade com o SQL transacional e ACID na memória, adicionando uma interface relacional em uma camada de dados in-memory.

VoltDB: Projetado por vários pesquisadores de sistema de banco de dados bem conhecidos, esse banco *oferece a velocidade e a alta escalabilidade dos bancos de dados NoSQL, mas com garantias ACID*, e sua latência em milissegundo e *integração com Hadoop*.

SQLFire: Servidor de *banco de dados NewSQL da VMware*, desenvolvido para escalar em plataformas nas nuvens e tomar as vantagens de infraestrutura virtualizadas.

MariaDB: foi *desenvolvido pelo criador do MySQL* e é totalmente compatível com o MySQL. Também pode interagir com os bancos de dados NoSQL, como Cassandra e LevelDB.

Algunes Bancos NewSQL

IGTI



Conclusão



- ✓ Definição e visão de NewSQL
- ✓ Exemplo de bancos de dados NewSQL

Próxima Aula



01. ••

03. ••

02. ••

04. ••

Técnicas dos bancos NoSQL

Armazenamento de Dados

Capítulo 02 – Aula 02.04 – Técnicas em Bancos NoSQL

PROF.: RICARDO BRITO ALVES

Nesta Aula



- ❑ Técnicas utilizadas na implementação de bancos
NoSQL

Técnicas Utilizadas na Implementação



Sobre as principais características nos bancos de dados **NoSQL**, é importante *ressaltar algumas técnicas utilizadas para a implementação de suas funcionalidades*.

Entre elas estão:

- Map/reduce
- Consistent hashing
- MVCC - Multiversion Concurrency Control
- Vector Clocks

Map Reduce

Permite a manipulação de enormes volumes de dados ao longo de nós em uma rede.

Funciona da seguinte forma:

- Na fase **MAP**, os *problemas são particionados em pequenos problemas que são distribuídos em outros nós na rede.*
- Quando chegam à fase **REDUCE**, esses pequenos problemas *são resolvidos em cada nó filho e o resultado é passado para o pai*, que sendo ele consequentemente filho, repassaria para o seu, até chegar à raiz do problema.

O que é Hashing?

Hashing é o processo de mapear um dado, normalmente um objeto de tamanho arbitrário para outro dado de tamanho fixo, normalmente um inteiro, conhecido como código hash ou simplesmente hash. Uma função é geralmente usada para mapear objetos para código hash conhecido como função hash.

Por exemplo, uma função hash pode ser usada para mapear strings de tamanho aleatório para algum número fixo entre 0... N. Dado qualquer string, ela sempre tentará mapeá-la para qualquer número inteiro entre 0 e N. Suponha que N seja 100. Então, por exemplo, para qualquer string, a função hash sempre retornará um valor entre 0 e 100.

Hello ---> 60

Hello World ---> 40

Consistent Hashing

Hashing consistente é um tipo especial de hashing, que *quando uma tabela hash é redimensionada, apenas as chaves precisam ser remapeadas em média, onde é o número de chaves e é o número de slots.*

Suporta mecanismos de armazenamento e recuperação, onde a quantidade de sites está em constante mudança. *É interessante usar essa técnica, pois ela evita que haja uma grande migração de dados entre estes sites, que podem ser alocados ou desalocados para a distribuição dos dados.*

Distributed Hashing

Suponha que um número de funcionários continue crescendo e se torne difícil armazenar todas as informações dos funcionários em uma tabela hash que pode caber em um único computador. Nessa situação, tentaremos distribuir a tabela de hash para vários servidores para evitar a limitação de memória de um servidor. Os objetos (e suas chaves) são distribuídos entre vários servidores.

Este tipo de configuração é muito comum para caches na memória como Memcached, Redis etc.

Distributed Hashing

A chave de partição também não deve ser confundida com uma chave primária, é mais como um identificador único controlado pelo sistema.

NAME	AGE	CAR	GENDER
jim	36	camaro	M
carol	37	345s	F
johnny	12	supra	M
suzy	10	mustang	F

Distributed Hashing

O banco de dados atribui um valor hash a cada chave de partição:

PARTITION KEY	MURMUR3 HASH VALUE
jim	-2245462676723223822
carol	7723358927203680754
johnny	-6723372854036780875
suzy	1168604627387940318

Distributed Hashing

Cada nó no cluster é responsável por um intervalo de dados com base no valor de hash.

Valores hash em um cluster de quatro nós:

NODE	START RANGE	END RANGE	PARTITION KEY	HASH VALUE
1	-9223372036854775808	-4611686018427387904	johnny	-6723372854036780875
2	-4611686018427387903	-1	jim	-2245462676723223822
3	0	4611686018427387903	suzy	1168604627387940318
4	4611686018427387904	9223372036854775807	carol	772335892720368075

MVCC - Multiversion Concurrency Control



Oferece suporte a transações paralelas em banco de dados. Por não fazer uso de locks para controle de concorrência, faz com que transações de escrita e leitura sejam feitas simultaneamente.

Ao serem iniciados novos processos de leitura de um banco de dados, e nesse mesmo instante existir um outro processo que está atualizando, pode acontecer que o processo de leitura esteja lendo veja apenas uma parte do que está sendo atualizado, ou seja, um dado inconsistente.

Vector Clocks

Vector clocks: *Ordenam eventos que ocorreram em um sistema.* Como existe a possibilidade de várias operações estarem acontecendo simultaneamente, o uso de *um log de operações informando suas datas se faz importante para informar qual versão de um dado é a mais atual.*

Conclusão



Técnicas utilizadas na implementação de bancos NoSQL:

- ✓ Map/reduce
- ✓ Consistent hashing
- ✓ MVCC - Multiversion Concurrency Control
- ✓ Vector Clocks

Próxima Aula



01. ••

Tipo de Bancos NoSQL

02. ••

03. ••

04. ••

Armazenamento de Dados

Capítulo 02 – Aula 02.05 – Tipo de Bancos NoSQL

PROF.: RICARDO BRITO ALVES

Nesta Aula

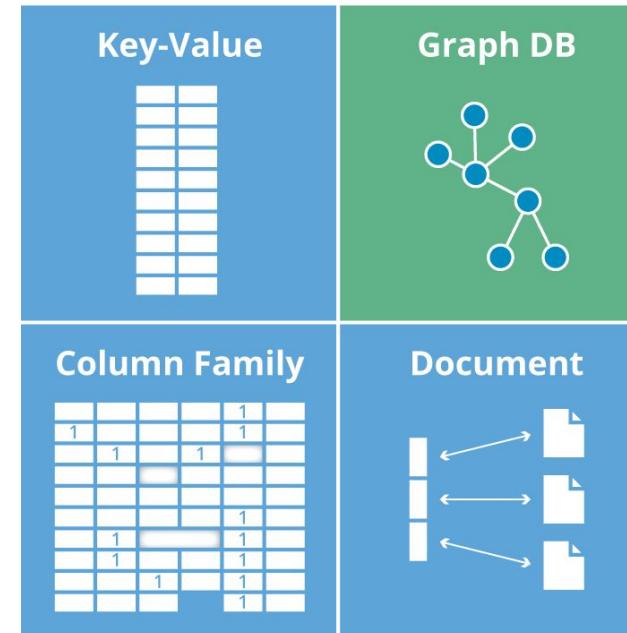


- ❑ Tipo de banco de dados NoSQL

Modelos de NoSQL

Temos quatro categorias do NoSQL:

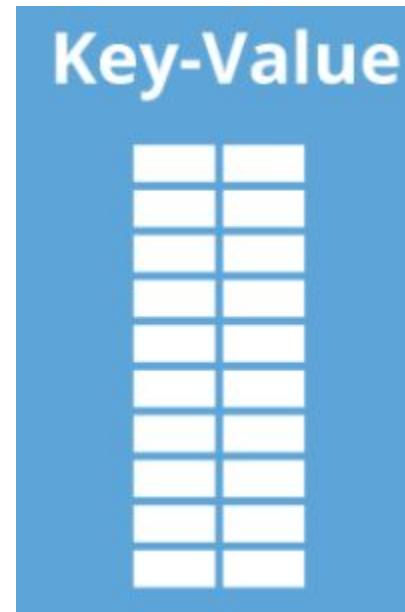
- Chave-valor (key-Value)
- Orientado a Grafos
- Orientado a Coluna (Column Family)
- Orientado a Documentos



Chave-Valor (Key-Value)

- Modelo mais simples.
- Permite a visualização do banco como uma grande tabela.
- Todo o banco é composto por um conjunto de chaves que estão associadas a um único valor.

Chave (Campo)	Valor (Instancia)
Nome	Hélio Rodrigues
Idade	45
Sexo	Masculino
Fone	99 99999999



Chave-Valor (Key-Value)

É um modelo considerado simples e permite a sua visualização através de uma tabela de hash, no qual há uma chave única e um indicador de determinado dado, podendo ser uma String ou um binário.

Este modelo é caracterizado pela sua facilidade ao ser implementado, permitindo que os dados sejam acessados rapidamente através da chave, aumentando também a disponibilidade do acesso aos dados.

Para manipulá-los, utilizamos comandos simples como **get()** e **set()**, que retornam e capturam valores.

Um problema enfrentado por este tipo de banco de dados é **que não permite a recuperação de objetos através de consultas mais complexas**.

Como exemplo, podemos citar o Dynamo que foi desenvolvido pela Amazon.

Orientado a Grafos

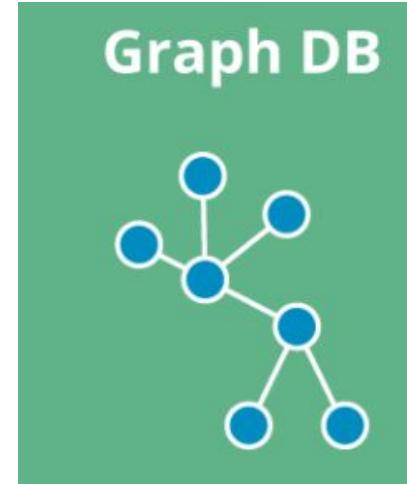
Este modelo possui três componentes básicos:

- Nós (vértices dos grafos).
- Os relacionamentos (arestas).
- As propriedades (conhecidos também como atributos).

É visto como um multigrafo rotulado e direcionado, onde cada par de nós **pode ser conectado por mais de uma aresta.**

A utilização deste modelo é muito útil quando é necessário fazer consultas demasiadamente complexas.

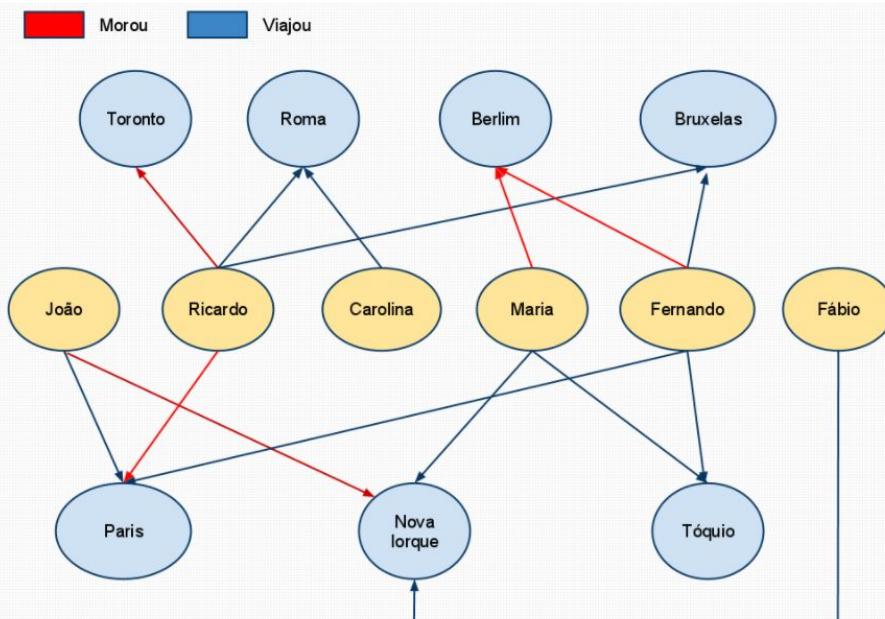
O modelo orientado a grafos possui uma alta performance, permitindo um bom desempenho nas aplicações.



Orientado a Grafos

Imagine uma aplicação que mantêm informações relativas à viagem. Uma consulta pertinente seria: “Quais cidades foram visitadas anteriormente por pessoas que foram para Nova Iorque?”

Temos diversas pessoas: João, Ricardo, Carolina, Maria, Fernando e Fábio que representam nós do grafo e estão conectadas a cidades que visitaram ou residiram.



Orientado a Grafos

O Neo4J trata-se de um banco de dados baseado em grafos desenvolvido em Java.

Além de possuir suporte completo para transactions, ele também trabalha com nós e relacionamentos.



Orientado a Colunas

Este tipo de banco de dados *foi criado para armazenar e processar uma grande quantidade de dados distribuídos em diversas máquinas.*

Aqui existem *as chaves, mas neste caso, elas apontam para atributos ou colunas múltiplas.*

Os dados são indexados por uma tripla (coluna, linha e timestamp), *a coluna e linha são identificadas por chaves e o timestamp permite diferenciar múltiplas versões de um mesmo dado.*

Como o próprio nome sugere, as colunas *são organizadas por família da coluna.* Demonstra maior complexidade que o de chave-valor.

Column Family			
1	1	1	1
1	1	1	
			1
1		1	1
1		1	1
	1	1	1
			1



Orientado a Colunas

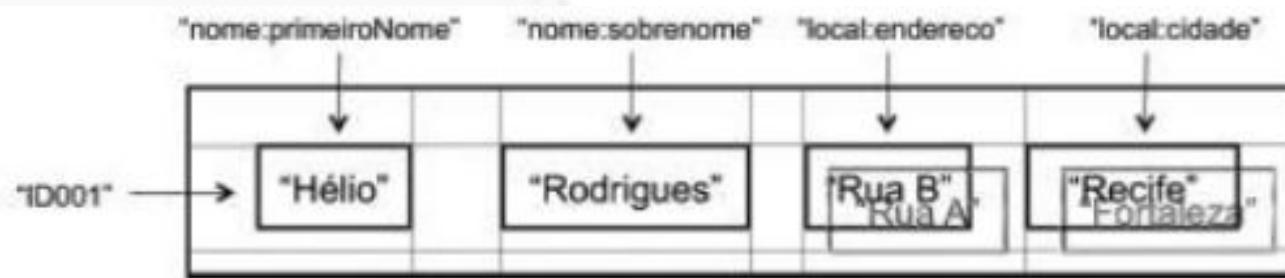
Vale destacar que as operações de escrita e leitura são atômicas, ou seja, os valores associados a uma linha são considerados em sua execução, independente das colunas que estão sendo lidas/escritas.

O conceito associado a este modelo é o de família de colunas, com o objetivo de reunir colunas que armazenam o mesmo tipo de informação.

Orientado a Colunas

Podemos modelar o conceito de amigos, onde o **primeiro nome** e **sobrenome** são colunas pertencentes à família de colunas denominada “**nome**”. Da mesma forma, as colunas **endereço**, **cidade** pertencem à **família local**.

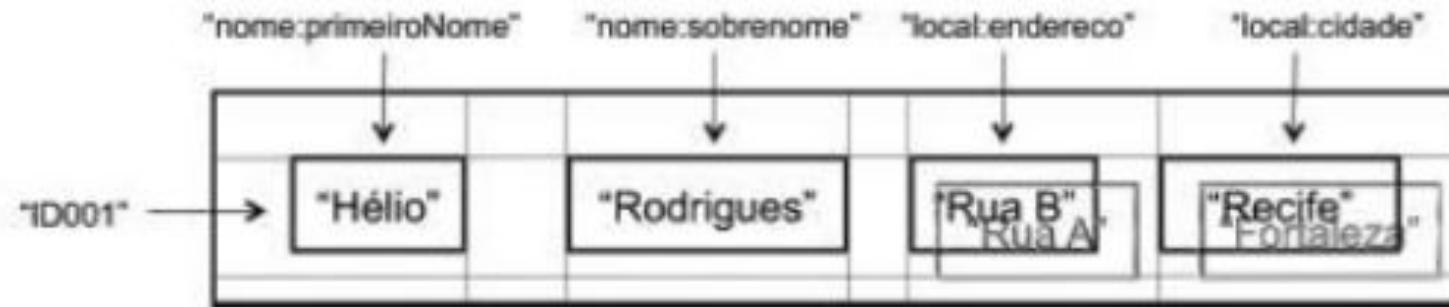
É interessante observar que na linha 001 a pessoa tem diversos endereços. Como a busca neste tipo de banco de dados é atômica, mesmo que o interesse seja buscar o primeiro nome da linha 001, todas as colunas serão retornadas quando esta mesma linha for consultada.



Orientado a Colunas

Este modelo permite ainda o particionamento de dados, oferecendo forte consistência, ***no entanto, a alta disponibilidade é o seu ponto fraco.***

Este modelo de dados surgiu com o BigTable criado pelo Google. Além do BigTable temos também o Cassandra que foi desenvolvido pelo Facebook.

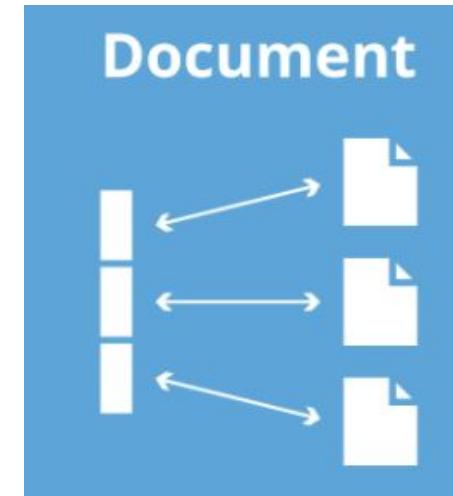


Orientado a Documentos

Como o próprio nome sugere, **este modelo armazena coleções e documentos**.

Um documento é um objeto identificador único e um conjunto de campos que podem ser strings, listas ou documentos aninhados.

Diferente do banco de dados chave-valor onde se cria uma única tabela hash, **neste modelo temos um agrupamento de documentos**, sendo que em cada um destes documentos temos um conjunto de campos e o valor destes campos.



Orientado a Documentos

Neste modelo temos ausência de esquema pré-definido (schema free).

Portanto é possível que haja alterações nos documentos, com a adição de novos campos, por exemplo, sem afetar adversamente outros documentos.

Outra característica interessante é que não é necessário armazenar valores de dados vazios para campos que não possuem um valor.

ID: P001
Assunto: "Eu gosto
Autor: "Hélio"
Data: "27/01/2011
Tags: ["laranjas", "s
Mensagem: "Hoje e

ID: P002
Assunto: "Eu gosto de tomates"
Autor: "Mara"
Data: 15/05/2012
Tags: ["tomate", "suco", "plantas"]
Mensagem: "Hoje eu estou com vontade de tomar suco de tomate."

Orientado a Documentos

Como exemplo de sistema de banco de dados que utiliza este tipo de solução destacamos o **CouchDB** e o **MongoDB**. O CouchDB utiliza o formato JSON e é implementado em Java. Já o mongo usa um formato semelhante ao JSON e é implementado em C++, **permitindo tanto concorrência quanto replicação**.

JSON

É um acrônimo de **JavaScript Object Notation**, e tem um formato compacto, de padrão aberto independente, de troca de dados simples e rápida entre sistemas, especificado por Douglas Crockford em 2000, que utiliza texto legível a humanos, no formato atributo-valor.

```
1 | {  
2 |     "id":1,  
3 |     "nome":"Alexandre Gama",  
4 |     "endereco":"R. Qualquer"  
5 | }
```

Modelos de NoSQL

Nenhum modelo pode ser considerado superior a outro. Um modelo pode ser mais adequado para ser utilizado em certas situações.

- Para a utilização de um banco de dados de manipulação de dados que frequentemente serão escritos, mas não lidos (um contador de hits na Web, por exemplo), pode ser usado um banco de dados orientado a documento como o **MongoDB**.
- Já aplicativos que demandam alta disponibilidade, onde a minimização da atividade é essencial, podemos utilizar um modelo orientado a colunas como o **Cassandra**.
- Aplicações que exigem um alto desempenho em consultas com muitos agrupamentos podem utilizar um modelo orientado a **grafos**.

Conclusão



Tipo de banco de dados NoSQL:

- ✓ Chave-valor (key-Value)
- ✓ Orientado a Grafos
- ✓ Orientado a Coluna (Column Family)
- ✓ Orientado a Documentos

Próxima Aula



01. •

Motivações no uso de bancos
NoSQL

02. •

03. •

04. •

Armazenamento de Dados

Capítulo 02 – Aula 02.06 – Motivações no uso de NoSQL

PROF.: RICARDO BRITO ALVES

Nesta Aula



- ❑ Motivações no uso de banco de dados NoSQL

NoSQL- Motivações



Hoje as empresas estão adotando NoSQL para um número crescente de cenários.

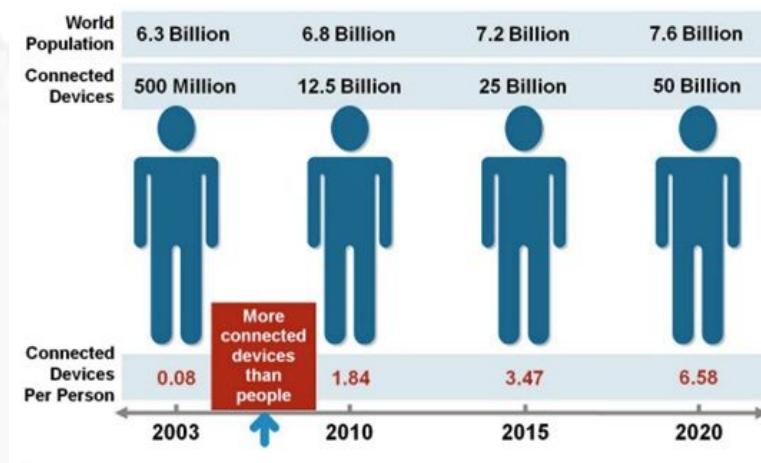
A escolha que é impulsionada por quatro tendências relacionadas :

- Big Users
- Big Data
- Internet das coisas
- Cloud Computing

NoSQL- Big Users

O crescente uso de aplicativos online resultou em um número crescente de operações de banco de dados e uma necessidade de uma maneira mais fácil de escalar bancos de dados para atender a essas demandas.

Um grande número de usuários, combinados com a natureza dinâmica dos padrões de uso está demandando uma tecnologia de banco de dados mais facilmente escalável.



NoSQL- Big Data



Big Data é a área do conhecimento que estuda como tratar, analisar e obter informações a partir de conjuntos de dados demasiadamente grandes para serem analisados por sistemas tradicionais.

Razões para usar o Big Data:

- Entender padrões;
- Prever situações;
- Criar fronteiras;
- Informar coleções de dados;
- Estimar parâmetros escondidos;
- Calibrar.

NoSQL- Big Data



Big Data é um conceito que **descreve o grande volume de dados estruturados e não estruturados** que são gerados a cada segundo.

O diferencial do Big Data está justamente atrelado à **possibilidade e oportunidade em cruzar esses dados por meio de diversas fontes** para obtermos insights rápidos e preciosos. A exigência dos consumidores e o aumento da competitividade em todos os mercados nos força a inovar e ter esse caminho como premissa básica nos negócios.



NoSQL- Tipos de Big Data

IGTI

Types of Big Data

Structured

1001 1010	1001 0101	1100 0110
0011 1100	0110 1001	0011 1010
0011 0011	0101 1100	1001 1001

Unstructured



Semi-Structured



SelectHub

NoSQL- IoT - Internet das Coisas



O termo foi utilizado pela primeira vez em 1999 para descrever um sistema onde os objetos poderiam ser conectados à internet.

Antigamente a internet ligava apenas computadores a computadores de uso exclusivo militar e de algumas poucas faculdades enquanto hoje em dia até um relógio pode ter acesso à internet. Além da necessidade de muito mais endereços de IP(IPv4 para IPv6 – 128 bits), essa revolução na internet traz muitas outras características pro meio.

O fato de ter tantos dispositivos conectados à rede literalmente tem revolucionado o modo como vivemos.

NoSQL- IoT - Internet das Coisas



- 32 bilhões de coisas vão estar conectadas a internet.
- 10% de todos os dados serão gerados por sistemas embarcados (versus 2% atualmente).
- 21% dos mais valiosos dados serão gerados por sistemas embarcados (versus 8% atualmente).
- Dados de telemetria - semi-estruturados e contínuos - representam um desafio para bancos de dados relacionais, que exigem um esquema fixo e dados estruturados.

Cidades Inteligentes

IGTI

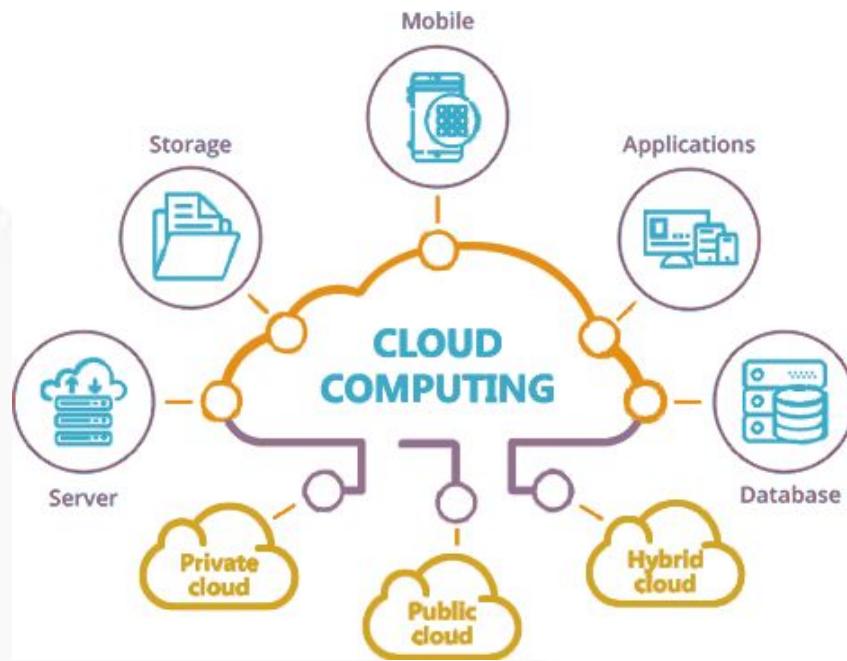
- Poluição Sonora.
- Otimizar coleta de lixo.
- Controle de tráfego.
- Controle de distribuição de energia elétrica.
- Segurança pública.



NoSQL- Cloud Computing



Computação em nuvem, é um termo coloquial para a disponibilidade sob demanda de recursos do sistema de computador, especialmente armazenamento de dados e capacidade de computação, sem o gerenciamento ativo direto do utilizador.



Conclusão



- ✓ Motivações no uso de bancos de dados NoSQL

Próxima Aula



01. ••

Principais bancos de dados
NoSQL

03. ••

02. ••

04. ••

Armazenamento de Dados

Capítulo 03 – Principais Banco de Dados NoSQL

PROF.: RICARDO BRITO ALVES

Armazenamento de Dados

Capítulo 03 – Aula 03.01 – Introdução ao Apache Cassandra

PROF.: RICARDO BRITO ALVES

Nesta Aula



- ❑ Apache Cassandra

Colunar

Para mostrar um banco de dados colunar, precisamos relembrar dos bancos relacionais: eles armazenam os dados em linhas (rows) enquanto os bancos colunares persistem os dados por colunas criando um relacionamento através de um id.

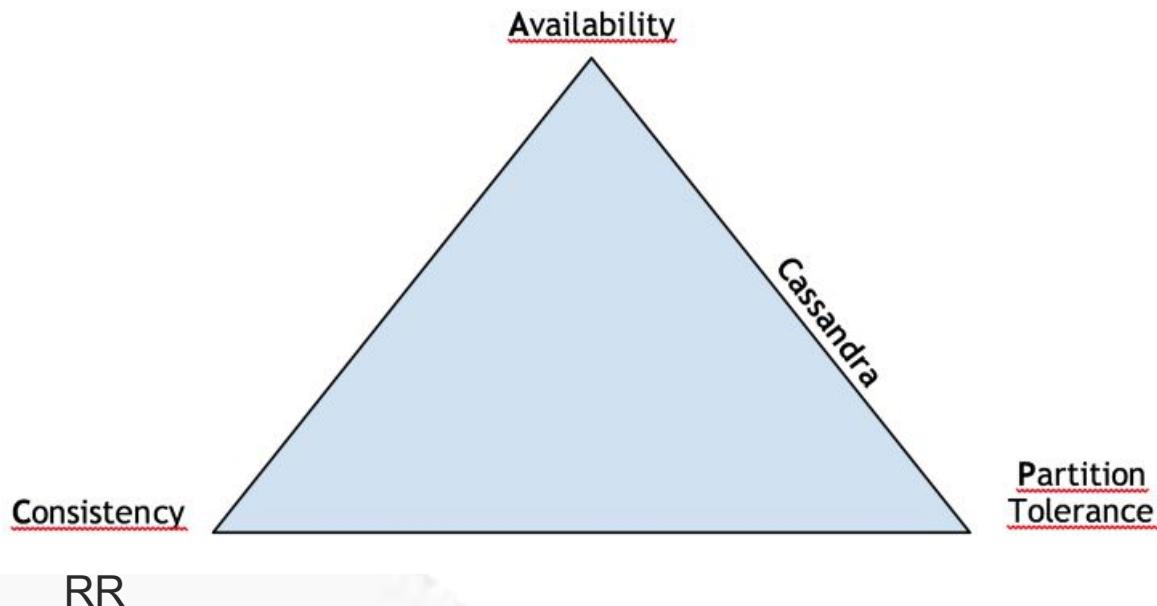
Banco de Dados Relacional:

sku_produto	nome_produto	preco_produto
1	Tênis	100
2	Bola	50
3	Camisa	200
4	Bike	900

Banco de Dados Colunar:

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

Teorema CAP



Resolvendo conflitos de forma transparente para os clients.

Apache Cassandra



Apache Cassandra é um projeto de sistema de banco de dados distribuído altamente escalável de segunda geração, que reúne a arquitetura do DynamoDB, da Amazon Web Services e modelo de dados baseado no BigTable, do Google.

Instalação Apache Cassandra



Faça o download da última versão no site oficial:

<http://incubator.apache.org/cassandra/>

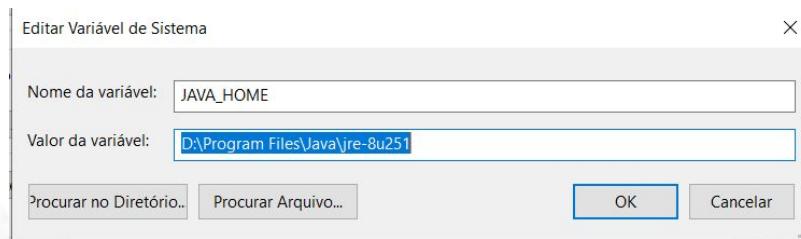
<https://downloads.apache.org/cassandra/3.11.9/apache-cassandra-3.11.9-bin.tar.gz>

- objetivo é fazer uma instalação bem simples para interação com o banco. Feito o download, descompacte o arquivo, os diretórios mais importantes são:
 - bin: contém os scripts de interação com o Cassandra.
 - data: contém os dados que o Cassandra armazena.
 - conf: contém os yml's de configuração do banco.

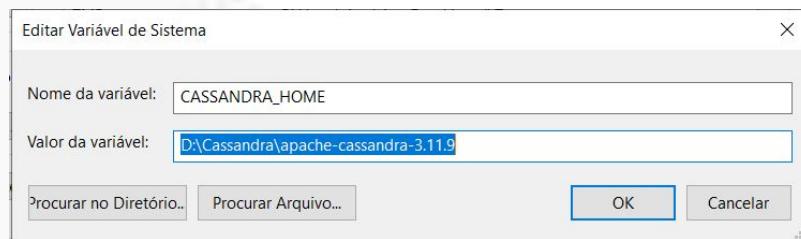
Instalação Apache Cassandra



A premissa é o ter o Java SDK ou JRE instalado na máquina. Como o Cassandra é feito em Java, o ideal é que você tenha a última versão do JRE instalado na sua máquina. Certifique que a variável de ambiente de sistema JAVA_HOME aponte para a instalação java da sua máquina.



Crie uma variável de ambiente chamada CASSANDRA_HOME e aponte para o seu diretório de instalação do Cassandra.



Rodando Apache Cassandra



Abra o command como administrador, acesse o diretório /bin onde
foi descompactado o Cassandra(cd\cassandra\apache-cassandra-3.11.9\bin). No
Windows é preciso liberar os comandos powershell no command.

- powershell Set-ExecutionPolicy Unrestricted
- cassandra.bat –f

Enquanto essa janela do command estiver aberta, o Cassandra
estará rodando localmente em apenas um nó e por default ele sobe
na porta: 9042.

Cqsh - Apache Cassandra



Neste momento vamos utilizar o **cqlsh**, é um sistema de command line que nos permite interagir com o Cassandra.

Abra uma nova janela do command como administrador, acesse o diretório /bin onde foi descompactado o Cassandra (cd\cassandra\apache-cassandra-3.11.9\bin) e execute os seguintes comandos:

- powershell Set-ExecutionPolicy Unrestricted
- cqlsh.bat

```
cqlsh:teste> insert into teste.posts (tag, name, author, description, likes)
... values
... ('apache-cassandra','Cassandra post','Jose','post do cassandra',1);
cqlsh:teste> update posts set likes = 2
... where tag = 'apache-cassandra' and name = 'Cassandra post';
cqlsh:teste> select * from posts;

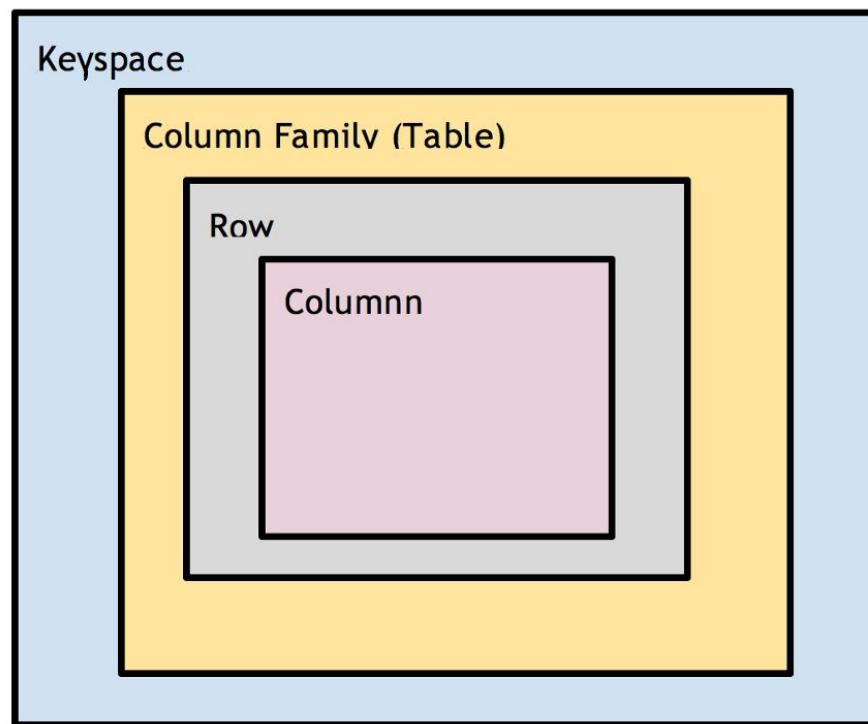
tag          | name        | author | description      | likes
-----+-----+-----+-----+-----+
apache-cassandra | Cassandra post | Jose | post do cassandra | 2

(1 rows)
cqlsh:teste>
```

Arquitetura do Cassandra

IGTI

O Cassandra é segmentado em keyspaces, tables (column families), rows e columns.



Keyspace

É o equivalente a um banco de dados no mundo relacional e agrupa as tabelas do sistema. Suas principais configurações são:

- Replication factor: O fator de replicação define quantas “cópias” existirão dos dados. Caso tenha um cluster com 5 máquinas e o replication factor igual a 2, o Cassandra irá garantir que seus dados estarão em pelo menos 2 máquinas. Essa feature é bem importante para alta disponibilidade do banco.
- Replica placement strategy: É a estratégia de replicação dos dados, existem 2 estratégias, sendo:
 - ✓ SimpleStrategy: Respeitando o fator de replicação o Cassandra simplesmente replica o dado para o próximo nó (node) do cluster.
 - ✓ NetworkTopologyStrategy: Com essa estratégia o Cassandra irá persistir os dados em mais de um datacenter (obrigatoriamente o cluster deve estar configurado para ser multi-datacenter).

Tables (Column family)

É o equivalente à uma tabela no mundo relacional.

Um keyspace pode conter “N” column families, que por sua vez podem conter “N” rows.

A comparação entre column families e tabelas do banco relacional é apenas para facilitar o entendimento.

Vale ressaltar que o comportamento é bem diferente, como por exemplo: JOIN`s não são suportados.

Tables (Column family)

Nas columns families são definidos os seguintes atributos:

- **Primary key:** A primary key é composta de 2 partes:
 - ✓ Partition key: Ela define qual partição será armazenado o dado.
 - ✓ Clustering key: É o restante da chave, esse campo não entra na definição das partições.

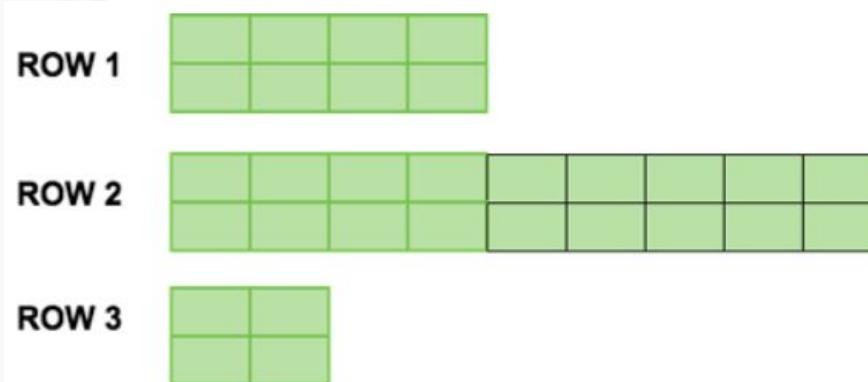
Obs: É possível definir “N” campos como partition key — a junção deles irá definir a partição.

- **Columns:** Representam as demais colunas da tabela, lista de tipos suportados na versão.

Row

As rows são compostas pela Primary key e um conjunto de columns.

Vale ressaltar que o Cassandra persiste apenas os campos que contém valores, diferente dos bancos relacionais onde são alocados recursos para campos nulos. Isso significa que as rows podem conter colunas diferentes.



Column

A column é composta por basicamente 3 campos:

- Column key: Nome da coluna.
- Column value: É o valor que está sendo persistido.
- Timestamp: O Cassandra utiliza esse campo para resolver conflitos e determinar qual é o valor mais atual.

Conclusão



- ✓ Apache Cassandra

Próxima Aula



01. ••

Introdução ao Redis

02. ••

03. ••

04. ••

Armazenamento de Dados

Capítulo 03 – Aula 03.02 – Introdução ao Redis

PROF.: RICARDO BRITO ALVES

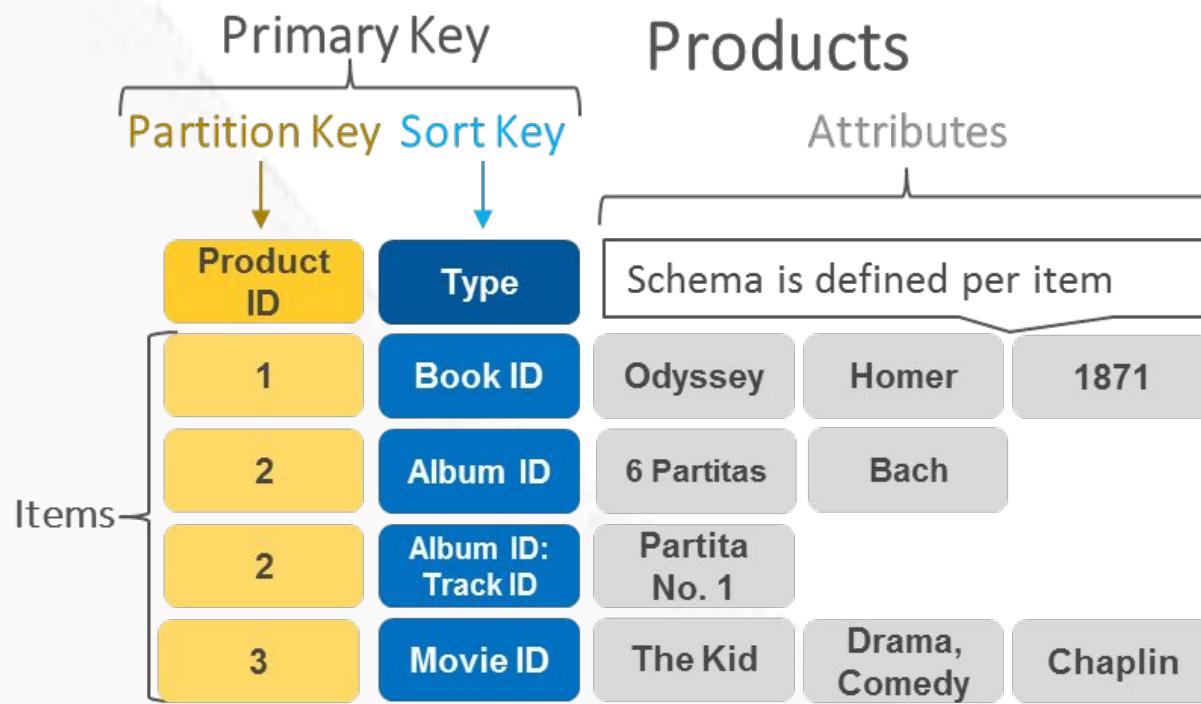
Nesta Aula



- ❑ Redis

Banco Chave-Valor

Banco de dados que trabalham no esquema Chave-Valor (Key-Value) são sistemas distribuídos, também conhecidos como tabelas de hash distribuídas, armazenam objetos indexados por chaves, e possibilitam a busca por esses objetos a partir de suas chaves.



Redis



Redis (Remote Dictionary Server) é um banco de dados de memória e de código aberto que é usado como cache e como intermediário de mensagens. Ele também é conhecido como um servidor de dados estruturados.

Oficialmente o Redis não tem uma distribuição para Windows, porém é possível encontrar uma distribuição paralela no GitHub.

Baixe o Redis do seu site oficial: <https://redis.io/>.

Redis

O Redis é um armazenamento de estrutura de dados de chave-valor de código aberto e na memória. O Redis oferece um conjunto de estruturas versáteis de dados na memória que permite a fácil criação de várias aplicações personalizadas. Os principais casos de uso do Redis incluem cache, gerenciamento de sessões, PUB/SUB e classificações. É o armazenamento de chave-valor mais conhecido atualmente.

- Desempenho muito rápido
- Estruturas de dados na memória
- Versatilidade e facilidade de uso
- Replicação e persistência
- Compatibilidade com as linguagens Java, Python, PHP, C, C++, C#, JavaScript, Node.js, Ruby, R, Go e muitas outras.

Redis - Comandos



<https://redis.io/commands>

Set: set name “José”

Get: get name

Keys*: busca todas as Keys setadas

del key: del name

Flushall: deleta todas as keys

Conclusão



- ✓ Redis

Próxima Aula



01. ••

Introdução ao Neo4j

02. ••

03. ••

04. ••

Armazenamento de Dados

Capítulo 03 – Aula 03.03 – Introdução ao Neo4j

PROF.: RICARDO BRITO ALVES

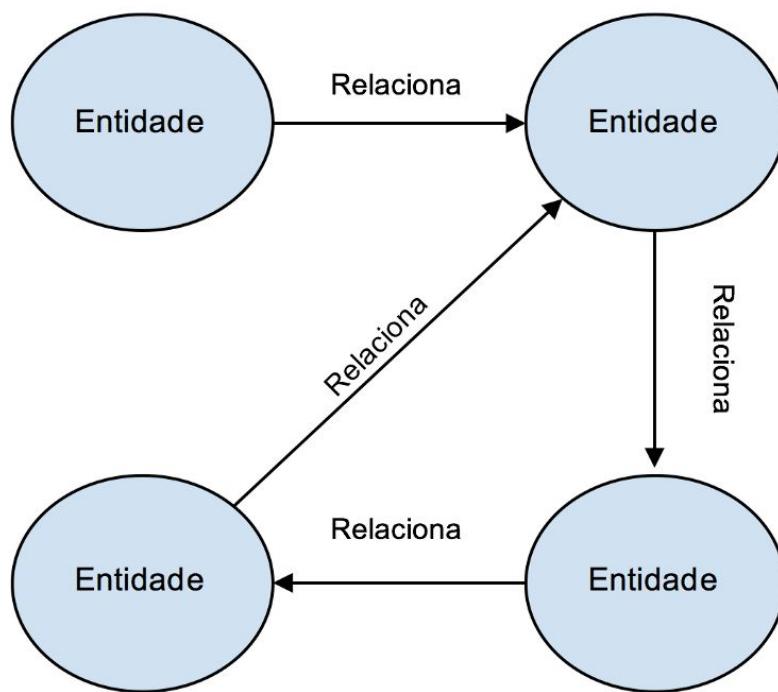
Nesta Aula



- Neo4j

Grafos

Os dados são persistidos em um esquema de grafo, onde um registro “aponta” para o próximo.



Neo4J



Neo4j é um sistema de gerenciamento de banco de dados gráfico desenvolvido pela Neo4j, Inc. Descrito por seus desenvolvedores como um banco de dados transacional compatível com ACID com armazenamento e processamento de gráfico nativo, Neo4j está disponível em uma "edição da comunidade" de código aberto.

O Neo4j é implementado em Java e acessível a partir de softwares escritos em outras linguagens usando a linguagem de consulta Cypher através de um endpoint HTTP transacional ou através do protocolo binário "bolt".

Neo4J



O Neo4J tem como prioridade, tratar seus relacionamentos da melhor forma possível, isso quer dizer que, na medida que os relacionamentos entre “nós” aumentam, sua capacidade de processamento continua estável, diferentemente de bancos de dados tradicionais.

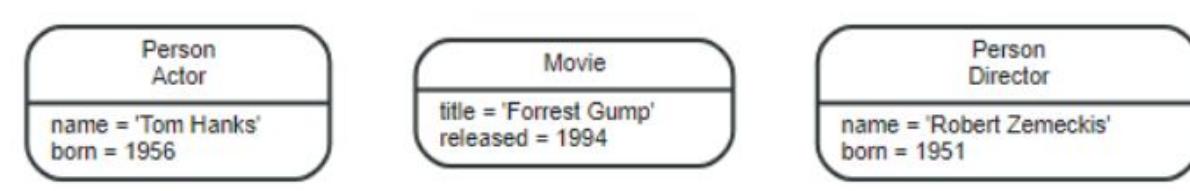
A linguagem do Neo4J é a **cypherQuery**, e a curva de aprendizagem não é tão grande, um case de sucesso é o facebook, que aderiu o sistema de grafos em um dos seus módulos, pois encontrou um desempenho maior para criar relacionamentos dentro da plataforma.

Neo4J

IGTI

Labels: É uma espécie de template para os nós, a grosso modo, seria uma classe. Na imagem, temos as labels:

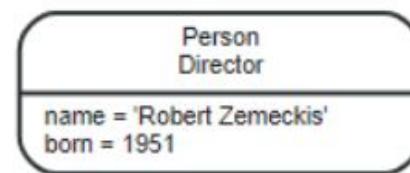
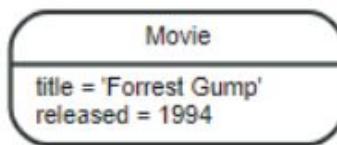
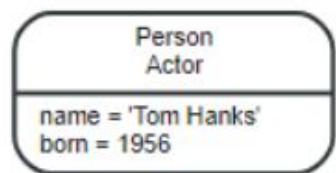
PersonActor, Movie e PersonDirector.



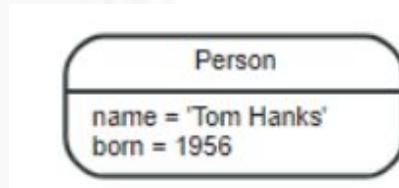
Neo4J

IGTI

Propriedades (Properties): são parâmetros baseados em chave-valor, para designar um nó, ou uma relacionamento. Na imagem há varias propriedades, tais como: name, born, title e released, eles demonstram características ou descrições de uma label.



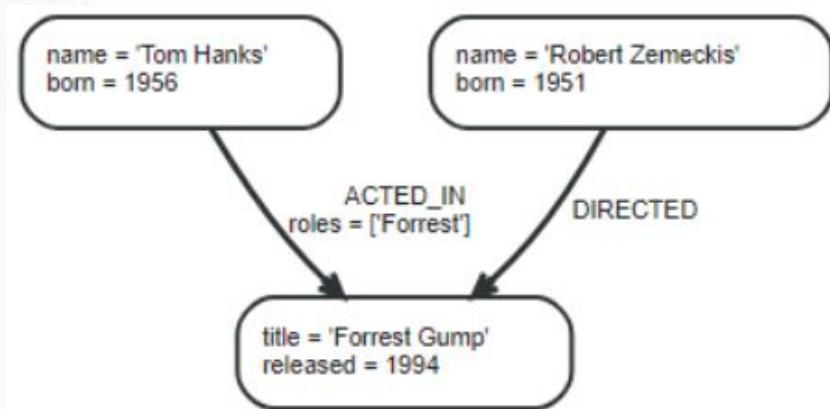
Instâncias: São como entidades criadas baseadas em um template pronto (label), se parecem instâncias de objetos de uma classe.



Neo4J

IGTI

Relacionamentos: Representa uma conexão entre nós, é possível criar propriedades dentro de um relacionamento. No exemplo abaixo, há dois relacionamentos, o primeiro se chama ACTED_IN, que possui uma propriedade chamada roles, o segundo relacionamento se chama DIRECTED, que não possui propriedades.



Instalação do Neo4J



Faça o download da versão que desejar no Neo4j Download Center.

<https://neo4j.com/download-center/>

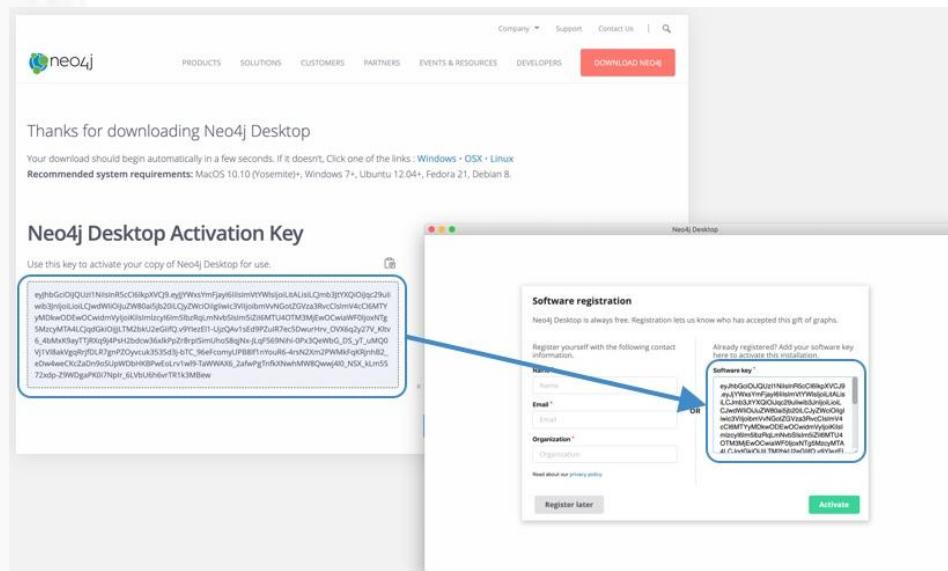
Current Releases

Enterprise Server	Community Server	Neo4j Desktop
Current Release		

Neo4J

IGTI

Se você estiver abrindo o Neo4j Desktop pela primeira vez, ele deverá solicitar que você registre o software com uma chave de ativação. Esta chave de ativação é gerada quando você faz o download do Neo4j Desktop pela primeira vez e será exibida na página de confirmação do download. Mantenha-o em um lugar seguro.

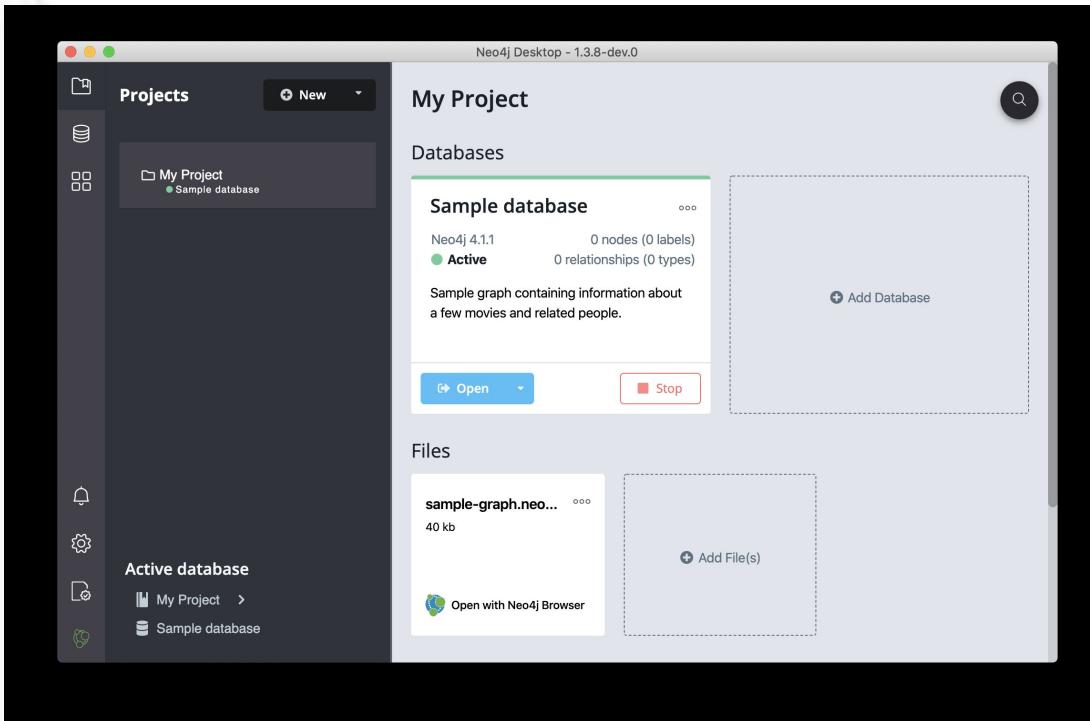


Neo4J

IGTI

Como um novo usuário do Neo4j Desktop, você será saudado com um novo banco de dados de amostra contendo Filmes e Atores.

<https://neo4j.com/developer/neo4j-desktop/>



Neo4J



É possível perceber a diferença entre os bancos relacionais e os bancos em grafo no que diz respeito à recuperação de informação, ao comparar como a mesma consulta é escrita para cada sistema.

Considere uma consulta que retorna todos os usuários que curtem a página com o nome “The Beatles”.

```
1  SELECT u.name, u.gender, u.age
2      FROM User u, Likes c, Page p
3      WHERE p.name = "The Beatles"
4          AND p.ID = c.ID_Page
5          AND c.ID_User = u.ID
```

[graph_rdbms.sql](#) hosted with ❤ by [GitHub](#)

[view raw](#)

```
1  MATCH (users) -[:LIKES]-> (:Page {name:"The Beatles"})
2  RETURN users
```

[graph_rdbms.cql](#) hosted with ❤ by [GitHub](#)

[view raw](#)

Neo4J



As palavras chave mais importantes do **Cypher** são:

MATCH: Busca no grafo, dados que correspondem a um padrão fornecido

WHERE: Filtra os resultados com predicados, como no SQL.

RETURN: Informa qual o retorno da sua consulta

CREATE: Cria elementos (nodes e relacionamentos) com rótulos e propriedades.

MERGE: É uma combinação de MATCH e CREATE.

Conclusão



- ✓ Neo4j

01. •

Introdução ao MongoDB

02. •

03. •

04. •

Armazenamento de Dados

Capítulo 03 – Aula 03.04 – Introdução ao MongoDB

PROF.: RICARDO BRITO ALVES

Nesta Aula

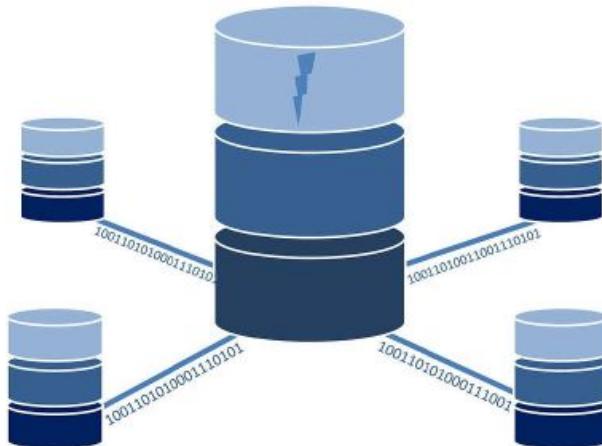


- ❑ MongoDB

NoSQL – Banco de Dados de Documentos



- Armazenam chave/valor.
- O valor é um documento estruturado e indexado, com metadados.
- Valor (Documento), pode ser consultado.
- JSON: JavaScript Object Notation.
 - Feito para troca de dados.
 - Mais compacto e legível que XML.



NoSQL - JSON

“ Estrutura nome/valor, entre aspas duplas



Dados separados por vírgula

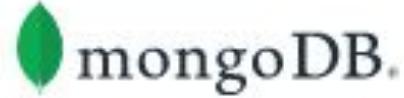


Chaves separam objetos



Vetores entre colchetes

MongoDB



MongoDB é um software de banco de dados orientado a documentos, de código aberto e multiplataforma, escrito na linguagem C++.

Classificado como um programa de banco de dados NoSQL, o MongoDB usa documentos semelhantes a JSON com esquemas.

MongoDB



- ✓ Open source.
- ✓ Multiplataforma.
- ✓ Escalável.



Relacional

Banco de Dados

Tabela

Linha

Coluna

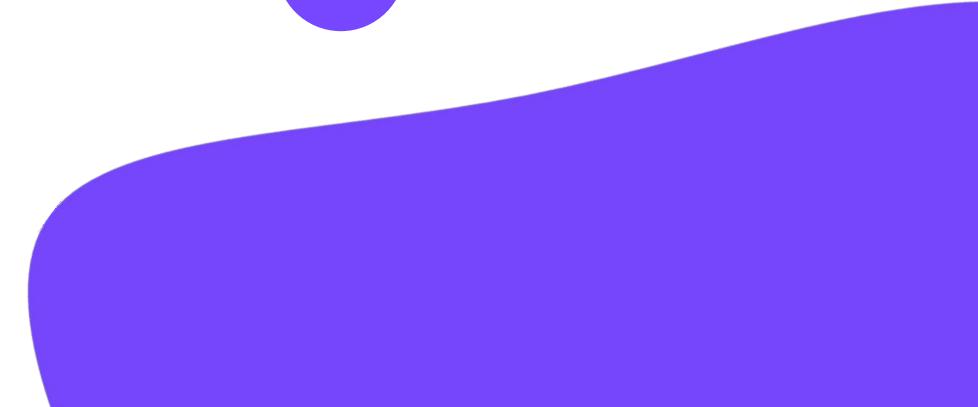
MongoDB

Banco de Dados

Coleção

Documento

Campo



Instalação MongoDB

Baixar a versão free Community no site:

<https://www.mongodb.com/try/download/community>

Windows: execute o instalador do MongoDB - clique duas vezes no arquivo “.msi” baixado.

Siga o assistente de instalação do MongoDB Community Edition. O assistente o orienta durante a instalação do MongoDB e MongoDB Compass.

Configuração de Serviço: à partir do MongoDB 4.0, você pode configurar o MongoDB como um serviço do Windows durante a instalação ou apenas instalar os arquivos e os iniciar.

MongoDB Compass



Interface gráfica do MongoDB.

The screenshot shows the MongoDB Compass interface. On the left, there's a sidebar with connection details: HOST localhost:27017, CLUSTER Standalone, and EDITION MongoDB 4.4.0 Enterprise. Below this is a search bar and a list of databases: DBElec, DBElections, DBEleicao, admin, config, and local. The main area displays a table of databases with columns: Database Name, Storage Size, Collections, and Indexes. Each database row has a trash icon in the last column.

Database Name	Storage Size	Collections	Indexes
DBElec	28.0KB	3	3
DBElections	4.0KB	1	1
DBEleicao	20.0KB	1	1
admin	20.0KB	0	1
config	24.0KB	0	2
local	44.0KB	1	1

Instalação MongoDB



MongoDB Service MongoDB

The following installs and configures MongoDB as a Windows service.

Starting in MongoDB 4.0, you can configure and start MongoDB as a Windows service during the install, and the MongoDB service is started upon successful installation.

This screenshot shows the 'Service Configuration' dialog box for MongoDB 4.0. The title bar reads 'MongoDB 4.0.0 2008R2Plus SSL (64 bit) Service Customization'. The main area is titled 'Service Configuration' with the sub-instruction 'Specify optional settings to configure MongoDB as a service.' Below this, there are two radio button options: 'Install MongoDB as a Service' (selected) and 'Run service as Network Service user' (unchecked). Under the local or domain user option, fields for 'Account Domain' (empty), 'Account Name' ('MongoDB'), and 'Account Password' (empty) are present. The 'Service Name' field is set to 'MongoDB'. At the bottom, the 'Data Directory' is set to 'C:\Program Files\MongoDB\Server\4.0\data\' and the 'Log Directory' is set to 'C:\Program Files\MongoDB\Server\4.0\log\'. Navigation buttons at the bottom include '< Back', 'Next >', and 'Cancel'.

Instalação MongoDB



The following installs MongoDB only and does not configure MongoDB as a Windows service.

If you choose not to configure MongoDB as a Windows service, uncheck the **Install MongoDB as a Service**.

Executando MongoDB

Execute o MongoDB – pelo command (execute como administrador) vá no diretório bin e execute o arquivo “mongo.exe”.

"C:\Program Files\MongoDB\Server\4.4\bin\mongo.exe"

```
D:\Program Files\MongoDB\Server\4.4\bin>mongo
MongoDB shell version v4.4.0
connecting to: mongodb://127.0.0.1:27017/?compressors=disabled&gssapiServiceName=mongodb
Implicit session: session { "id" : UUID("1843fa47-4730-4fae-9668-05bf5e03d650") }
MongoDB server version: 4.4.0
---
The server generated these startup warnings when booting:
    2020-11-22T11:03:56.207-03:00: ***** SERVER RESTARTED *****
    2020-11-22T11:03:59.973-03:00: Access control is not enabled for the database. Read and write access to data and
configuration is unrestricted
---
MongoDB Enterprise > ls collections
```

MongoDB - Definições

MongoDB é um banco de dados multi-plataforma orientado ao documento que fornece alta performance, alta disponibilidade e fácil escalabilidade.

MongoDB funciona no conceito de coleções de documentos.

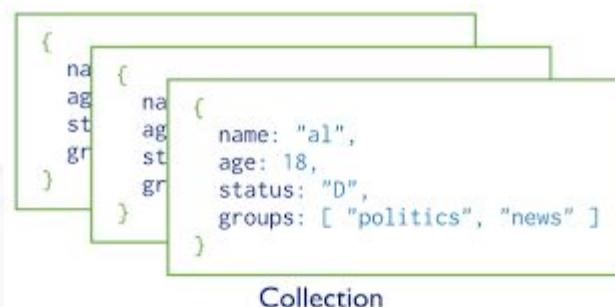
Banco de Dados

Banco de dados é um recipiente físico para coleções. Cada banco de dados tem o seu próprio conjunto de arquivos no sistema de arquivos. Um único servidor MongoDB normalmente tem vários bancos de dados.

MongoDB - Collection

Coleções (Collection)

- Collection é um grupo de documentos MongoDB.
- É o equivalente de uma tabela de RDBMS.
- Uma coleção existe dentro de um único banco de dados. Coleções não impõem um esquema.
- Documentos dentro de uma coleção pode ter diferentes campos. Normalmente, todos os documentos em uma coleção são propositalmente semelhantes ou afins.



MongoDB - Document

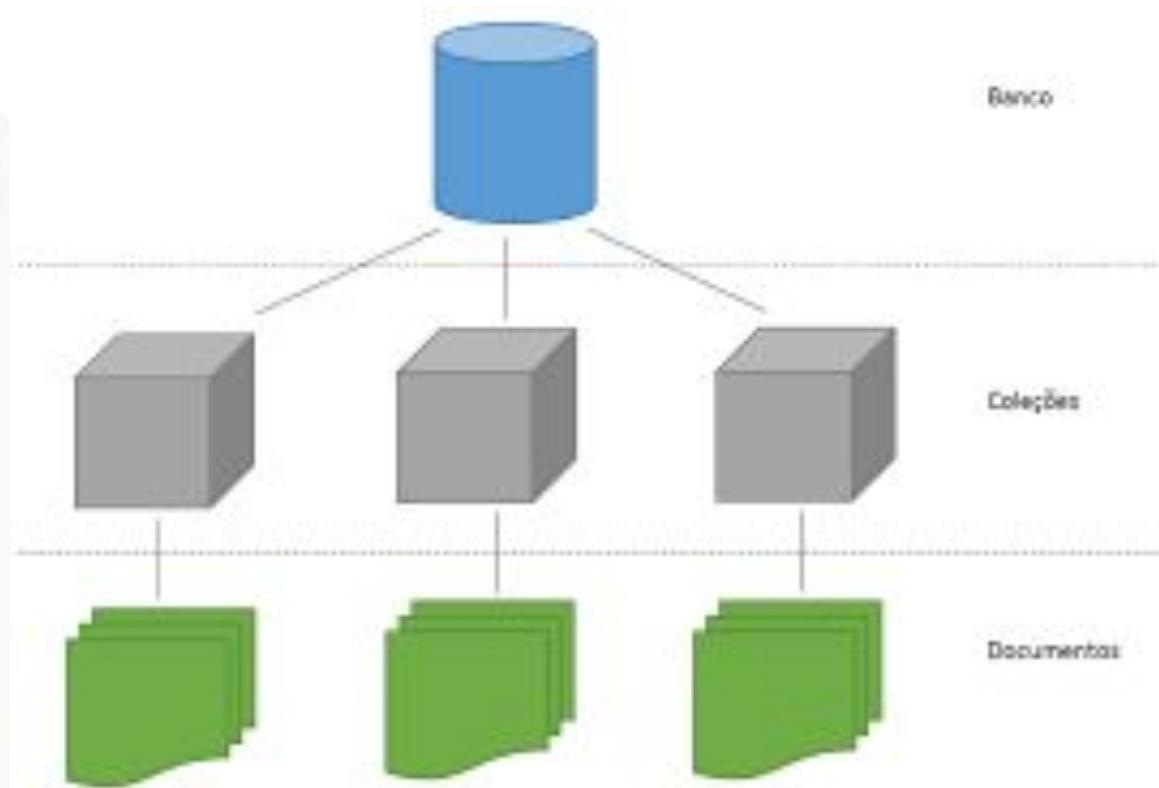
Document

- Um documento é um conjunto de pares de valores-chave.
- Documentos têm esquema dinâmico. Esquema dinâmico significa que os documentos na mesma coleção não precisam ter o mesmo conjunto de campos ou estrutura e campos comuns em documentos de uma coleção podem conter diferentes tipos de dados.

```
db.users.insertOne(  
  {  
    name: "sue",      ← field: value  
    age: 26,         ← field: value  
    status: "pending" ← field: value  
  })                ← collection
```

MongoDB - Definições

IGTi



MongoDB - Comandos

<https://docs.mongodb.com/manual/reference/command/>

```
db.posts.insert([
  {nome:"André",postagem:"Produtos caros", data:"12-01-2019",idade:25},
  {nome:"Ricardo",postagem:"Produtos caros", data:"14-07-2019", idade:12}])

#idade menor ou igual a 12
> db.posts.find({postagem:"Produtos caros",idade: {$lte: 12}})
{ "_id" : ObjectId("5d0911600ee1100c307004da"), "nome" : "Ricardo", "postagem"
: "Produtos caros", "data" : "14-07-2019", "idade" : 12 }
```

- **\$lt: menor que**
- **\$lte: menor ou igual que**
- **\$ne: diferente de**
- **\$in: contém**
- **\$nin: não contém**

MongoDB - Comandos



```
> db.posts.find()

{ "_id" : ObjectId("5d090bc10ee1100c307004d4"),
  "nome" : "José", "postagem" : "Bons Produtos!",
  "data" : "31-06-2019" }

{ "_id" : ObjectId("5d090cd10ee1100c307004d5"),
  "nome" : "Antonio", "postagem" : "Minha bike
quebrou", "data" : "26-05-2019" }

{ "_id" : ObjectId("5d090cd10ee1100c307004d6"),
  "nome" : "Maria Silva", "postagem" : "Encontrei
tudo que procurava", "data" : "14-06-2019" }

{ "_id" : ObjectId("5d090cd10ee1100c307004d7"),
  "nome" : "Lucas Andrade", "postagem" : "Ótimo
atendimento!", "data" : "12-04-2019" }
```

Conclusão



- ✓ MongoDB

Próxima Aula



01.

03.

Prática com MongoDB

02.

04.

Armazenamento de Dados

Capítulo 03 – Aula 03.05 – Prática com MongoDB

PROF.: RICARDO BRITO ALVES

Nesta Aula



- Prática com MongoDB

MongoDB – Create Database

MongoDB Create Database - criar um banco de dados MongoDB dinamicamente.

O comando MongoDB **USE DATABASE** é usado não apenas para **selecionar um banco de dados para executar consultas**, mas **também para criar um banco de dados**. Se o nome do banco de dados fornecido para o comando USE Database ainda não estiver presente no MongoDB, um novo banco de dados com o nome será criado quando você inserir um Documento em uma Coleção nesse banco de dados.

```
use <database_name>
```

MongoDB – Create Database



```
> show dbs;  
  
> use teste_aula  
  
> show dbs;  
  
> db.users.insertOne( { name: "Foo", age: 34, cars: [ "BMW  
320d", "Audi R8" ] } )  
  
> db.users.insertOne( { name: "TFC-test", age: 31, cars: [ "BMW  
320d", "Audi R8" ] } )  
  
> show dbs;
```

MongoDB – Delete Database

Para excluir ou descartar um banco de dados do MongoDB, siga as etapas abaixo:

Selecione o banco de dados que deseja excluir com a ajuda do comando USE <database>.

Elimine o banco de dados com a ajuda do comando db.dropDatabase () .

```
> show dbs  
  
> use teste_aula  
  
> db.dropDatabase()  
  
> show dbs
```

MongoDB – Collection

- A Collection é um armazenamento para documentos.
- É análogo a uma tabela no MySQL que armazena registros.
- Ao contrário dos bancos de dados relacionais, o MongoDB não tem um esquema. Um documento no MongoDB em uma collection pode ter quaisquer campos. Portanto, mudanças no documento não afetam documentos anteriores armazenados na mesma collection.

MongoDB – Collection

MongoDB – Create Collection

```
db.createCollection(name, options)
```

name	[mandatory] name of collection
options	[optional] mongodb document specifying information about collection

Options

Field	Type	Description
capped	Boolean	If true then collection is limited in size. i.e., Number of documents that could be stored in the collection is limited.
autoIndexId	Boolean	[deprecated]
size	Number	Size of Collection in Bytes
max	Number	Number of MongoDB Documents that could be stored in the collection

MongoDB – Collection

MongoDB – Create Collection

```
> use aula  
  
> show collections  
  
> db.customers.insert({ name: "Honey", age: 25, cars:  
    [ "Audi R8" ] })  
  
> show collections
```

MongoDB – Collection

MongoDB – Delete Collection

> use aula

> show collections

> db.customers.drop()

> show collections

Se a coleção não existe o MongoDB retorna false

> db.exemplo.drop()

MongoDB – Documents

- **Documents** em MongoDB é uma entidade na qual zero ou mais pares de valores de campo ordenados são armazenados.
- Em comparação com bancos de dados relacionais, é análogo a um registro ou linha na tabela.
- O documento no MongoDB segue as especificações BSON [<http://bsonspec.org/>]. BSON é a serialização com codificação binária de documentos semelhantes a JSON. Com o BSON, os documentos do MongoDB podem ser acessados facilmente. Como BSON usa tipos de dados C, codificar dados para BSON ou decodificar de BSON é mais fácil na maioria das linguagens de programação.

MongoDB – Documents

Um documento pode ter documentos aninhados nele. Os documentos do MongoDB são os blocos de construção de uma coleção do MongoDB.

```
{  
  name: "Midhuna",  
  age: 23,  
  place: "New York",  
  hobbies: ["Singing", "Reading Books"]  
}
```

```
{  
  name: "Midhuna",  
  age: 23,  
  place: "New York",  
  hobbies: ["Singing", "Reading Books"]  
  spouse: {  
    name: "Akash",  
    age: 25  
  }  
}
```

MongoDB – Insert Documents

```
> use aula  
  
> db.customers.insertOne( { name: "Abhi", age: 34, cars: [  
  "BMW 320d", "Audi R8" ] } )
```

MongoDB – Insert Documents



```
db.customers.insertMany(  
  [  
    { name: "Midhuna", age: 23, cars: [ "BMW 320d", "Audi R8" ],  
      place:"Amaravati" },  
    { name: "Akhil", age: 24, cars: [ "Audo A7", "Agera R" ],  
      place:"New York" },  
    { name: "Honey", age: 25, cars: [ "Audi R8" ] }  
    { name: "Paul", age: 25, cars: [ "Audi R8" ] }  
    { name: "Stuart", age: 25, cars: [ "Audi R8" ] }  
  ]  
)
```

MongoDB – Query Documents



`db.<collection>.find({})` – recupera todos documentos de
uma collection

> `use aula`

> `db.customers.find({})`

> `criteria={ age:23 }`

> `db.customers.find(criteria)`

MongoDB – Query Documents



```
db.collection.find(query_document, projection_document)
```

```
{field1:projection_value, field1:projection_value, ...}
```

Projection_value	Description
1	Include the field:value in Result
0	Do not include the field:value in Result

- use aria

```
> projection_doc={"name":1,"age":1,"cars":1,_id:0}
```

```
> db.customers.find({},projection_doc)
```

```
> db.customers.find({}, {"name":1, "age":1})
```

```
> db.customers.find({}, {"name":0, "age":0})
```

MongoDB – Update Documents



db.collection_name.update(criteria, update, options)

collection_name	Name of the collection in which you update document
criteria	[mandatory] Criteria to select documents for update
update	[mandatory] Updates to the fields for the selected documents

Option	[Type][DefaultValue] DefaultAction
upsert	[boolean][false] Does not insert a new document when no match found
multi	[boolean][false] Updates only one document.
collation	[document][default] Uses binary comparison for comparing strings.
WriteConcern	[document][default] Refer Write Concern [https://docs.mongodb.com/manual/reference/write-concern/].

MongoDB – Update Documents



```
> db.customers.find({})  
  
> criteria={name:"Akhil"}  
  
> db.customers.find(criteria)  
  
> update={name:"Akhil xxx",age:28}  
  
> db.customers.update(criteria,update)  
  
> db.customers.find(criteria)
```

MongoDB – Update Documents



```
> criteria={age:25}  
  
> db.customers.find(criteria).count()  
  
> update={$set:{cars: "Audi R8 xxx" }}  
  
> options={multi:true}  
  
> db.customers.update(criteria,update,options)  
  
> db.customers.find(criteria)
```

MongoDB – Delete Documents



`db.collection_name.remove(CRITERIA, JUST_ONE)`

collection_name	String	[mandatory] Name of the Collection from which you would like to remove Documents
CRITERIA	Document	[optional] The criteria that selects the required documents to delete
JUST_ONE	1 or true	[optional] If one or true, deletes only one document from the selection

MongoDB – Limitando Documents

- MongoDB Limit Documents - Para limitar o número de registros que uma consulta retorna no resultado, use o método cursor.limit () .
- O método Limit aceita um número como argumento que indica o limite do número de registros no resultado da consulta.

> db.customers.find().limit(2)

- Skipping é uma forma de ignorar registros e geralmente é útil quando você já mostrou os primeiros N documentos e está interessado em mostrar apenas os documentos restantes.

> db.customers.find().skip(2)

MongoDB – Sort Documents

- MongoDB Sort Documents - Para classificar documentos em uma coleção com base em um campo, use o método cursor.sort () .
- O método de classificação aceita pares de campo e pedido em um documento como argumento.

```
cursor.sort({field:order, field:order [, field:order]}
```

Order	Description
1	Ascending (Increasing) Order
-1	Descending (Decreasing) Order

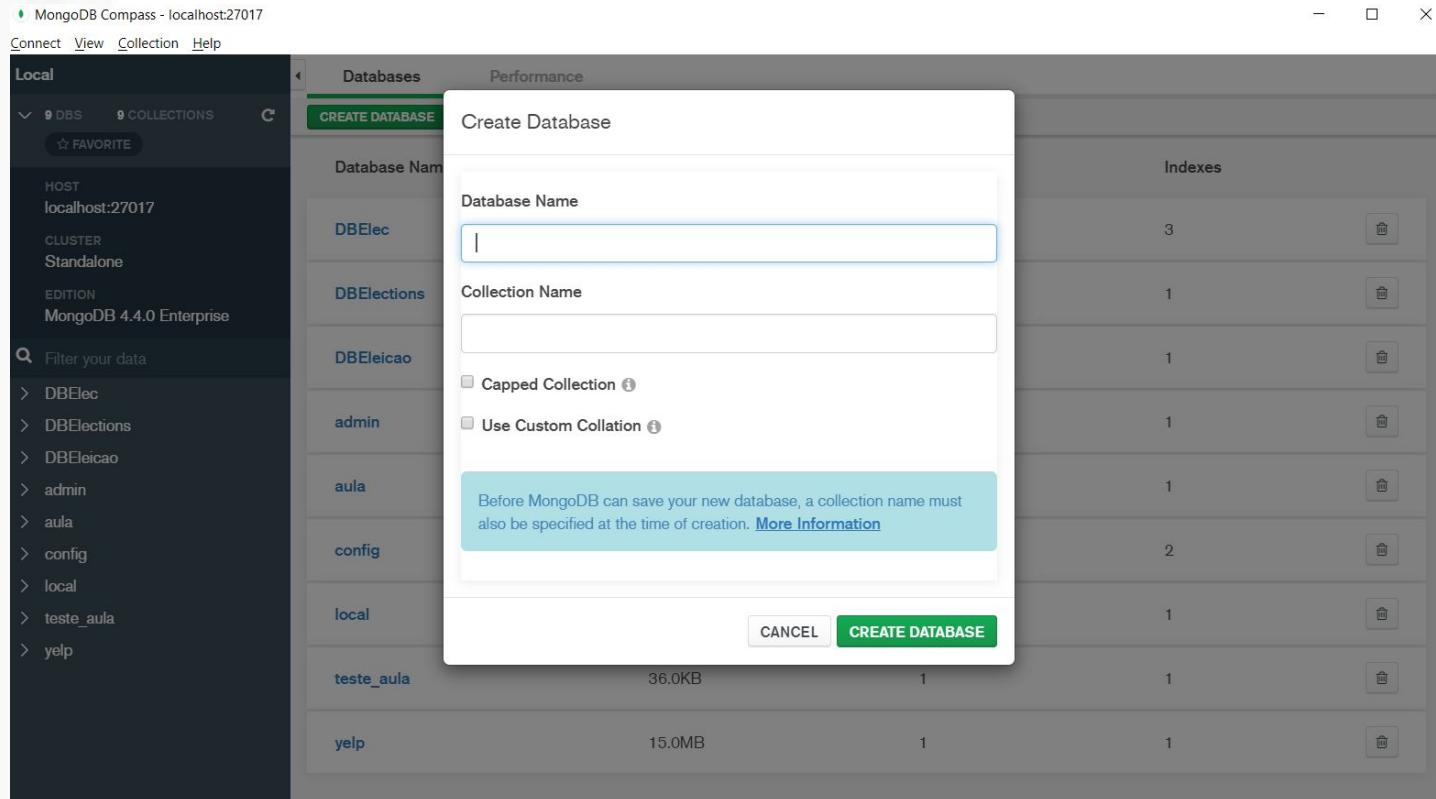
```
> db.customers.find().sort({"age":1});
```

```
> db.customers.find().sort({});
```

```
> db.customers.find().sort({"age": -1, "name": 1});
```

MongoDB – Import Compass

IGTI



Conclusão



- ✓ Prática com o banco de dados NoSQL MongoDB

Próxima Aula



01. •

Armazenamento de dados em
nuvem e sistemas de arquivos

02. •

03. •

04. •

Armazenamento de Dados

Capítulo 04 – Armazenamento de Dados em Nuvem e Sistemas de Arquivos

PROF.: RICARDO BRITO ALVES

Armazenamento de Dados

Capítulo 04 – Aula 04.01 – Introdução a Banco de Dados em
Nuvem

PROF.: RICARDO BRITO ALVES

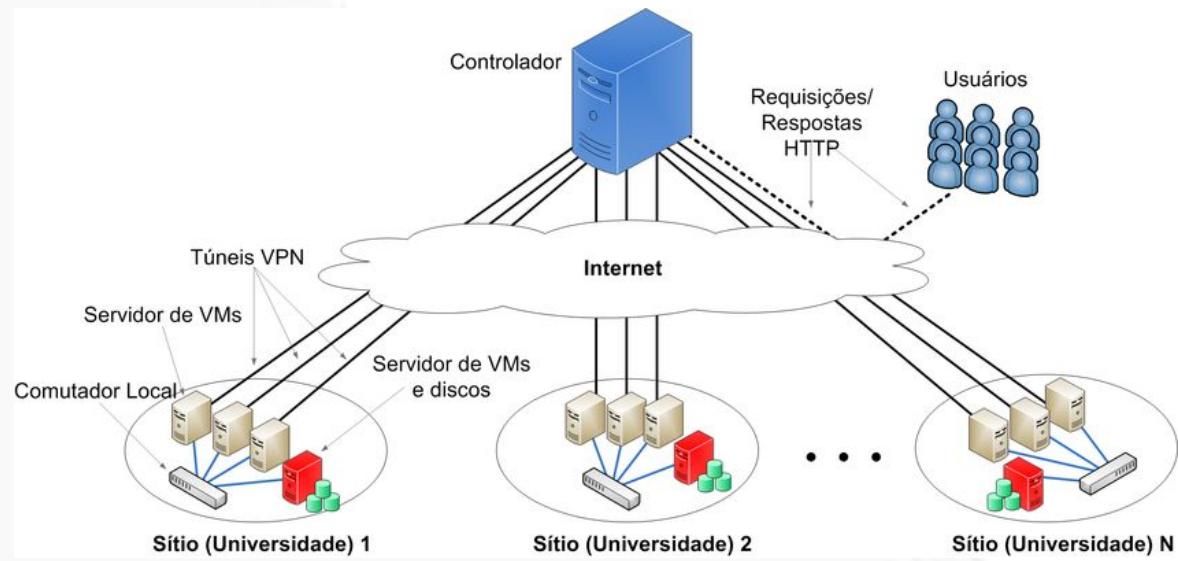
Nesta aula

- Modelos em Nuvem
- DBaaS
- Tipos de Nuvem
- Principais Fornecedores

Computação em Nuvem

IGTI

- Provisionamento dinâmico de recursos sob demanda
- Escalabilidade
- Cobrança baseada no uso do recurso ao invés de uma taxa fixa
- Distribuição geográfica dos recursos



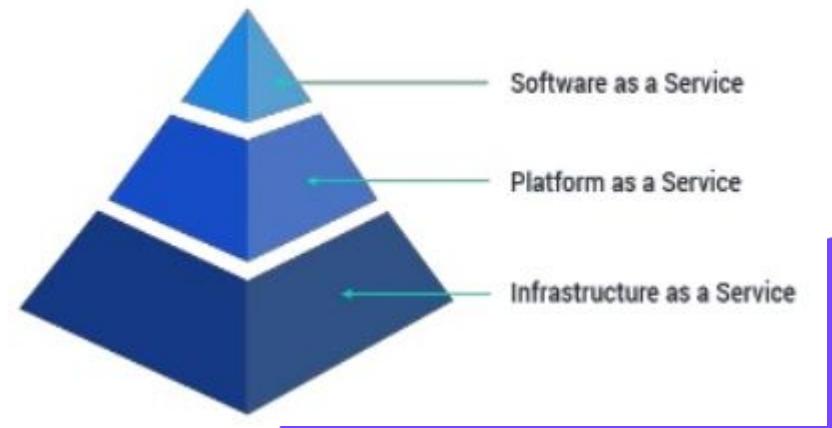
Modelos em Nuvem



SaaS é caracterizado principalmente pela não-aquisição de licenças de software.

PaaS envolve um ambiente virtual para criação, hospedagem e controle de softwares e bancos de dados.

IaaS apenas abstraí aspectos relacionados à parte física de servidores e redes.



IaaS — Infrastructure as a Service (Infraestrutura como Serviço)



Nesse exemplo dos modelos de nuvem, **a empresa contrata uma capacidade de hardware que corresponde a memória, armazenamento, processamento, etc.** Podem entrar nesse pacote de contratações os servidores, roteadores, racks, entre outros.

Dependendo do fornecedor e do modelo escolhido, a sua empresa pode ser tarifada, por exemplo, pelo número de servidores utilizados e pela quantidade de dados armazenados ou trafegados. **Em geral, tudo é fornecido por meio de um data center com servidores virtuais, em que você paga somente por aquilo que usar.**

PaaS — Platform as a Service (Plataforma como Serviço)



Nesse cenário, o PaaS surge como o ideal porque é, como o próprio nome diz, *uma plataforma que pode criar, hospedar e gerir esse aplicativo*.

Nesse modelo de nuvem, contrata-se um ambiente completo de desenvolvimento, no qual é possível criar, modificar e otimizar softwares e aplicações.

Aqui, a grande vantagem é que *a equipe de desenvolvimento só precisa se preocupar com a programação do software, pois o gerenciamento, manutenção e atualização da infraestrutura ficam a cargo do fornecedor e as várias ferramentas de desenvolvimento de software são oferecidas na plataforma*.

SaaS — Software as a Service (Software como Serviço)



Nesse modelo de nuvem **você pode ter acesso a um software sem comprar a sua licença, utilizando-o a partir da Cloud Computing**.

Muitos CRMs ou ERPs trabalham no sistema SaaS.

Assim, o acesso a esses softwares é feito usando a internet. Os dados, contatos e demais informações podem ser acessados de qualquer dispositivo, dando mais mobilidade à equipe.

O Google Docs e o Office 365 funcionam dessa maneira.

On-Premises

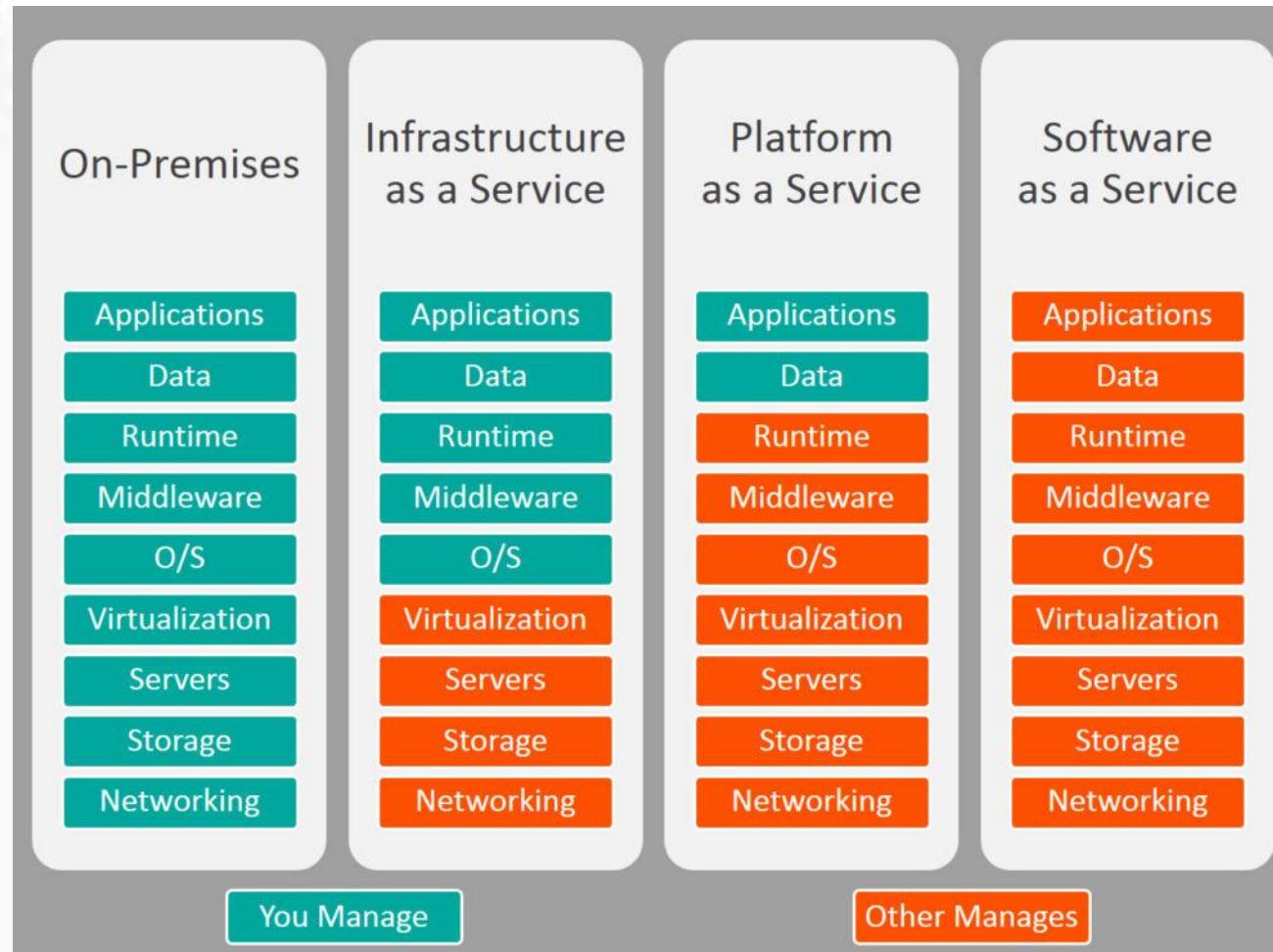


Um servidor ***on-premises*** é aquele em que a própria empresa tem a responsabilidade de processar suas aplicações de hardware e software.

Em outras palavras, toda a infraestrutura, customização, configuração e atualização é feita internamente.

Computação em Nuvem

IGTI



Banco de Dados em Nuvem



Vantagens

- Menor TCO (Total Cost of Ownership, significa Custo Total de Propriedades).
- Facilidade de manutenção.
- Maior elasticidade.
- Menor investimento inicial.

Desvantagens

- Maior percepção de custo.
- Segurança e privacidade (dados enviados para um provedor externo).

BDaaS - Banco de Dados Como Serviço



Virtualização: permite que banco de dados seja instalado em uma máquina virtual

DBaaS: fornece uma plataforma flexível escalável, sob demanda que está orientada para o autoserviço e gerenciamento fácil, particularmente em termos de provisionamento de um negócio no próprio ambiente

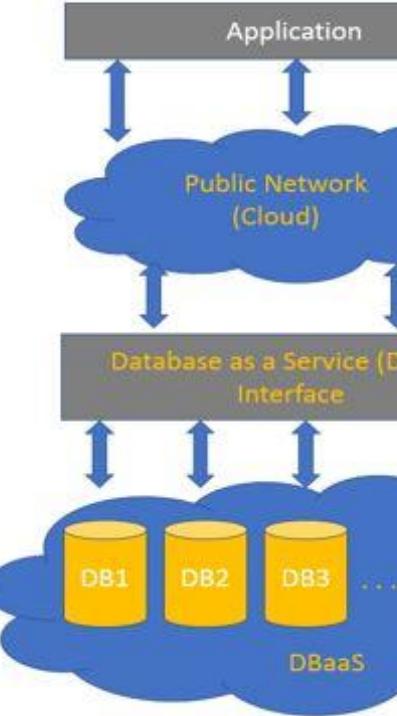


Banco de Dados Como Serviço

DBaaS oferece a funcionalidade de um banco de dados **semelhante ao que é encontrado em sistemas de gerenciamento de banco de dados relacionais (RDBMS), como o SQL Server, MySQL e Oracle.**

Sendo baseado em nuvem, por outro lado o **DBaaS fornece uma plataforma flexível escalável**, sob demanda que está orientada para o autoserviço e gerenciamento fácil, particularmente em termos de provisionamento de um negócio no próprio ambiente.

Produtos DBaaS normalmente fornecem capacidades de monitoramento suficientes para acompanhar o desempenho e o consumo e para alertar os usuários sobre possíveis problemas.



Cloudbase Database Arch

Banco de Dados Como Serviço



As *desvantagens* para o modelo DBaaS *incluem a falta de controle sobre os problemas de desempenho de rede, tais como falhas de latência e de aplicação inaceitáveis.*

Além disso, *alguns produtos DBaaS não suportam capacidades dos RDBMS típicos, tais como compressão de dados e partições de tabela.*

Antes de contratar um DBaaS, é essencial que a empresa avalie suas necessidades específicas e garanta que elas sejam satisfatoriamente resolvidas.

Principais Fornecedores

IGTI



Computação em Nuvem



Como se diferenciam:

	Nuvem pública	Nuvem privada	Servidores dedicados	Nuvem híbrida
Descrição	Ambiente multi-inquilino com escalabilidade de pagamento pelo uso.	Escalabilidade, mais a segurança e o controle aprimorados de um ambiente de inquilino único	Para cargas de trabalho previsíveis que exigem segurança e controle aprimorados	Conecte a nuvem pública à sua nuvem privada ou a servidores dedicados — até mesmo em seu próprio data center
Hardware físico	Compartilhado	Dedicada	Dedicada	Compartilhado + dedicado
Melhor para	Operações não sigilosas, voltadas para o público, e tráfego imprevisível	Operações sigilosas críticas	Operações sigilosas críticas e requisitos exigentes de desempenho, segurança e conformidade	Combine nuvem privada, pública e/ou servidores dedicados para o melhor de cada um deles

Computação em Nuvem

IGTI

Como se diferenciam (Rackspace)

	Nuvem pública	Nuvem privada	Servidores dedicados	Nuvem híbrida
Descrição	Ambiente multi-inquilino com escalabilidade de pagamento pelo uso.	Escalabilidade, mais a segurança e o controle aprimorados de um ambiente de inquilino único	Para cargas de trabalho previsíveis que exigem segurança e controle aprimorados	Conecte a nuvem pública à sua nuvem privada ou a servidores dedicados — até mesmo em seu próprio data center
Expansível	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>
Custo baixo, cobrança como serviços públicos	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>
Flexibilidade	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>
Personalizável	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
Alto desempenho	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
Segurança e controle aprimorados	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
Custo previsível	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>

Computação em Nuvem

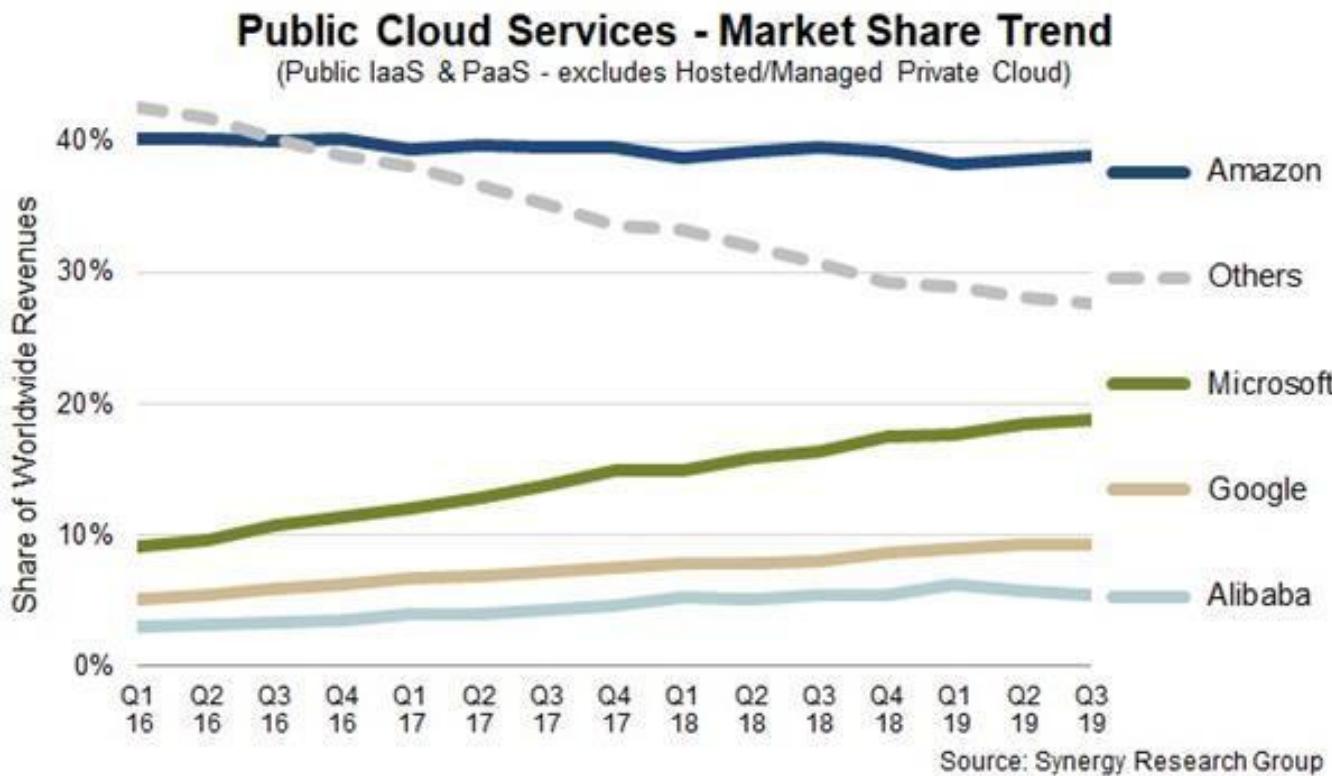


Nós temos três principais fornecedores de computação em nuvem, **AWS**, **Microsoft Azure** e **Google Cloud**, que possuem pontos fortes e fracos que os tornam ideais para diferentes cenários.



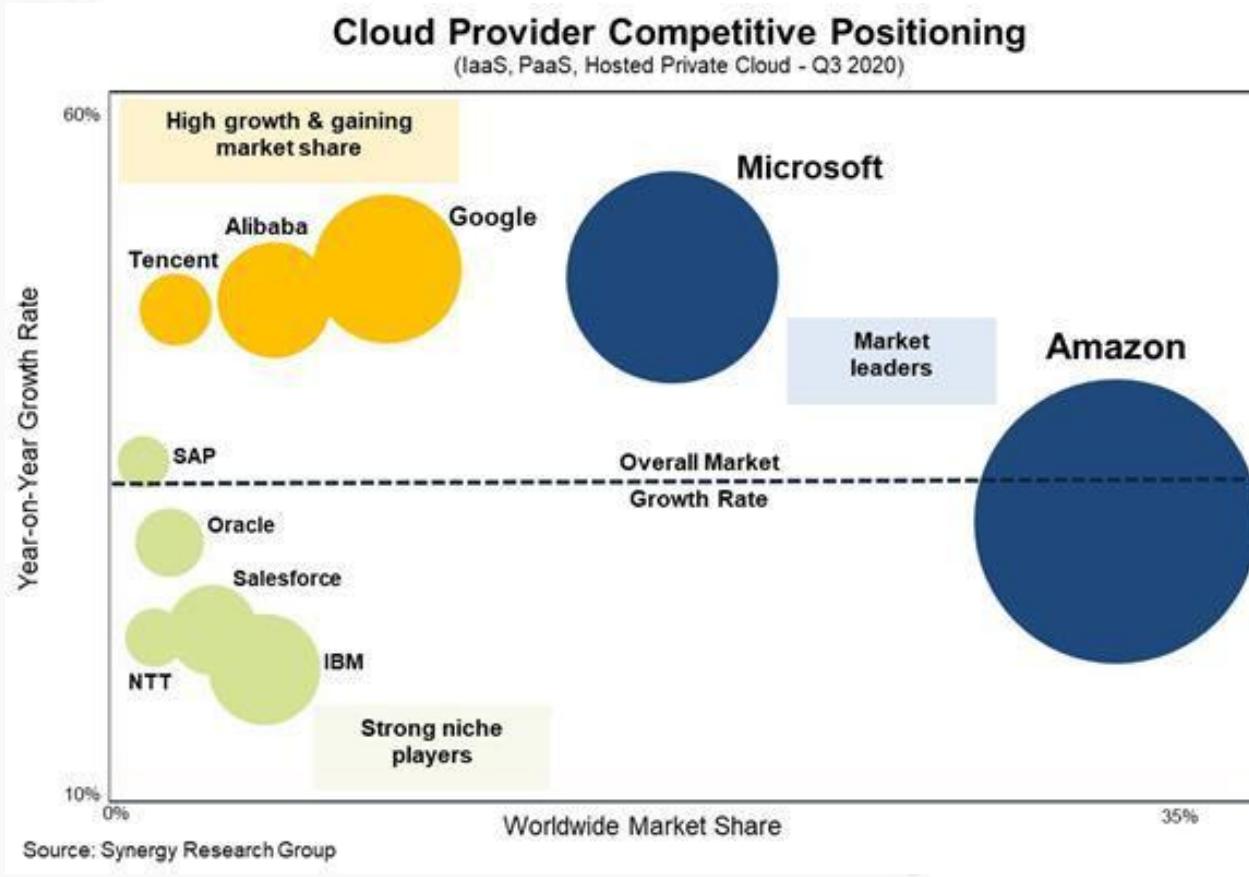
Synergy Research Group

IGTI



Synergy Research Group

IGTI



Synergy Research Group



APAC – Ásia Pacífico

Cloud Services Leadership – APAC Region

Rank	Total Region	China	Japan	Rest of East Asia	South & Southeast	Oceania
Leader	Amazon	Alibaba	Amazon	Amazon	Amazon	Amazon
#2	Alibaba	Tencent	Fujitsu	Microsoft	Microsoft	Microsoft
#3	Microsoft	Baidu	Microsoft	Alibaba	Google	Telstra
#4	Tencent	China Telecom	NTT	Google	Alibaba	Google
#5	Google	Sinnet-AWS	Google	Naver	NTT	Alibaba
#6	NTT	China Unicom	Softbank	KT	IBM	IBM

Based on IaaS, PaaS and hosted private cloud revenues in Q2 2020

Source: Synergy Research Group

Conclusão



Vimos os principais conceitos de nuvem.

- ✓ Modelos em Nuvem
- ✓ DBaaS
- ✓ Tipos de Nuvem
- ✓ Principais Fornecedores

Próxima Aula



01. ••

Introdução a AWS

02. ••

03. ••

04. ••

Armazenamento de Dados

Capítulo 04 – Aula 04.02 – Introdução a AWS

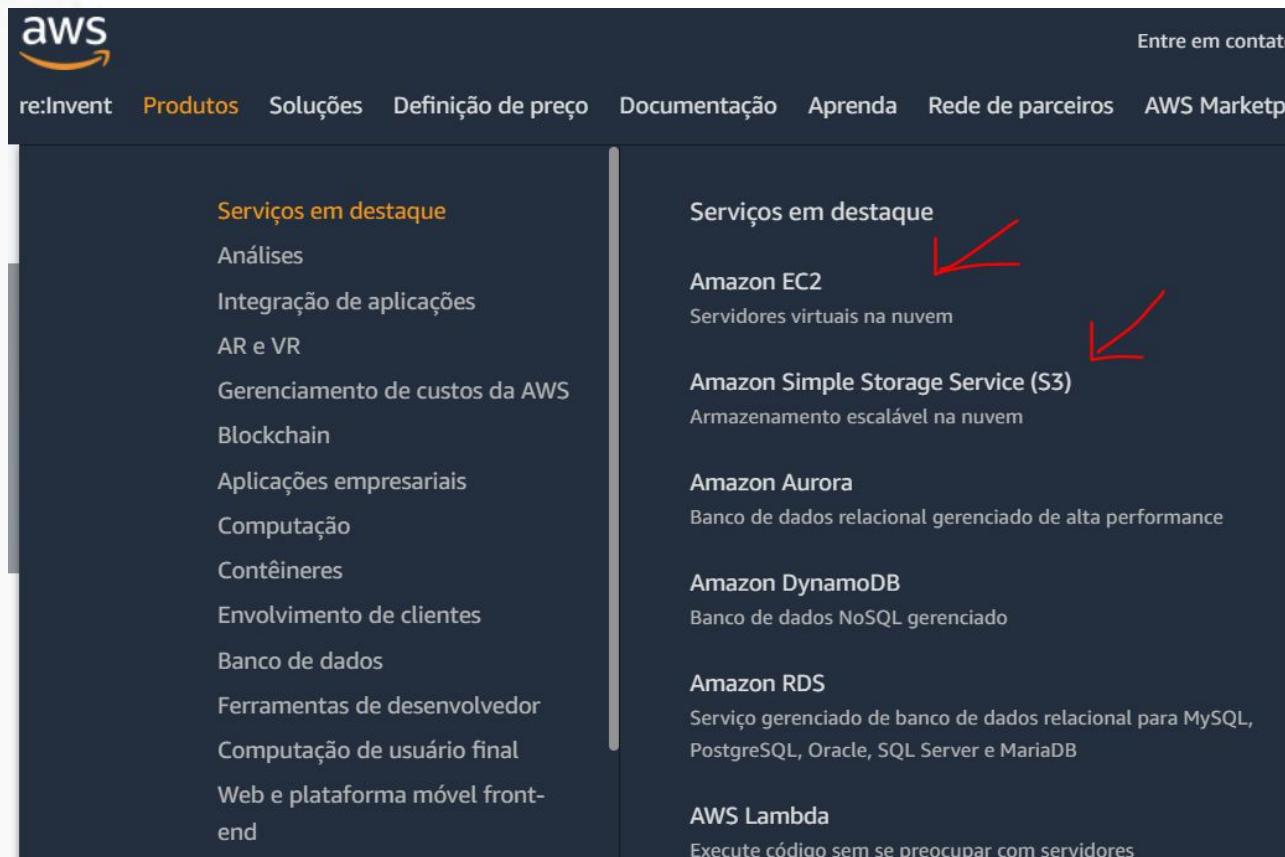
PROF.: RICARDO BRITO ALVES

Nesta aula



- AWS
- Principais Serviços

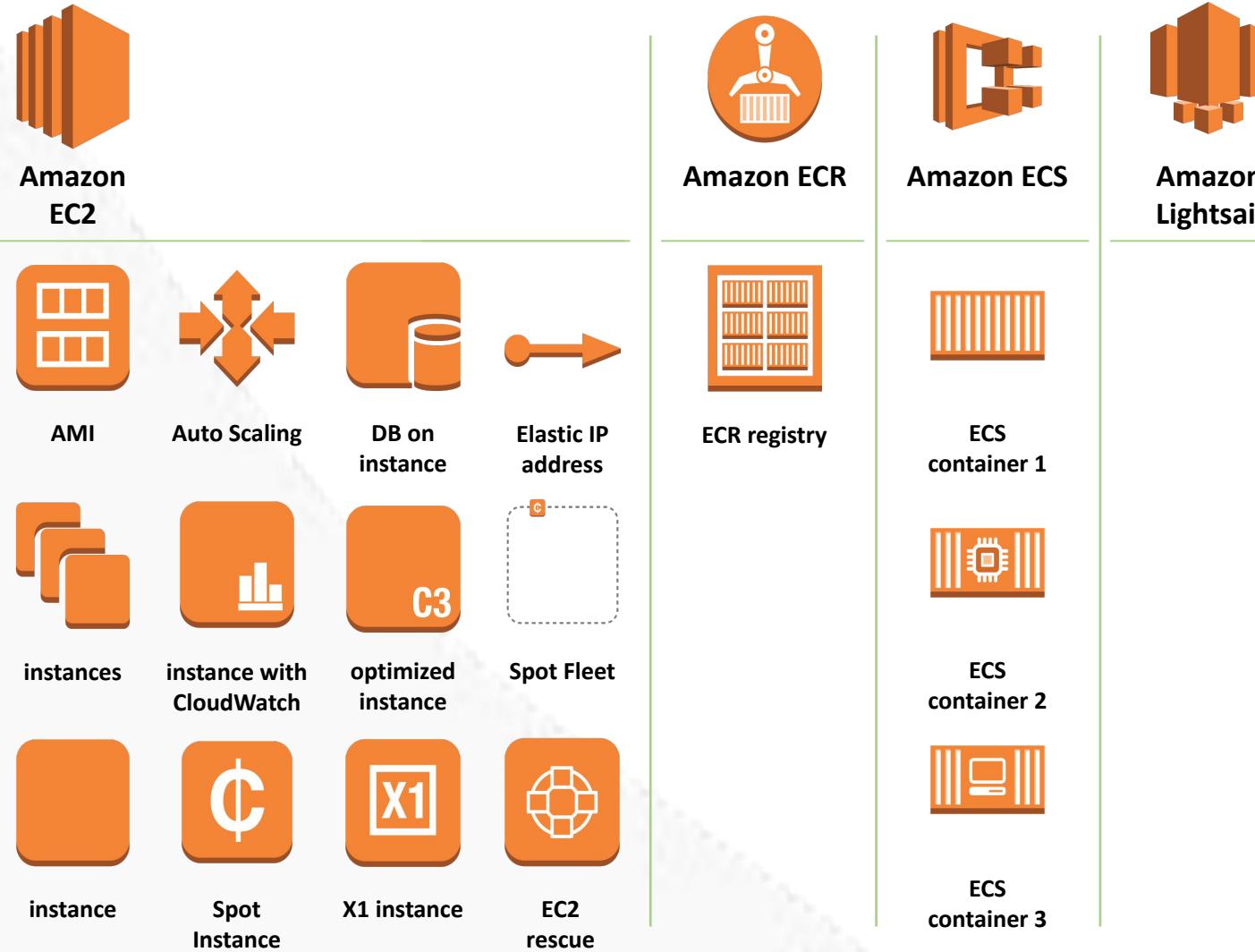
<https://aws.amazon.com/pt/getting-started/>



The screenshot shows the AWS Getting Started page. On the left, there's a sidebar with a list of services. On the right, there's a main content area with a list of six services, each with a brief description. Two red arrows point from the sidebar to the main content area, highlighting the first two services: Amazon EC2 and Amazon Simple Storage Service (S3).

Serviços em destaque	Serviços em destaque
Análises	Amazon EC2 Servidores virtuais na nuvem
Integração de aplicações	Amazon Simple Storage Service (S3) Armazenamento escalável na nuvem
AR e VR	Amazon Aurora Banco de dados relacional gerenciado de alta performance
Gerenciamento de custos da AWS	Amazon DynamoDB Banco de dados NoSQL gerenciado
Blockchain	Amazon RDS Serviço gerenciado de banco de dados relacional para MySQL, PostgreSQL, Oracle, SQL Server e MariaDB
Aplicações empresariais	AWS Lambda Execute código sem se preocupar com servidores
Computação	
Contêineres	
Envolvimento de clientes	
Banco de dados	
Ferramentas de desenvolvedor	
Computação de usuário final	
Web e plataforma móvel front-end	

Computacional



AWS – EC2



O Amazon *Elastic Compute Cloud* (Amazon EC2) oferece uma capacidade de computação escalável na Nuvem da Amazon Web Services (AWS).

O uso do Amazon EC2 ***elimina a necessidade de investir em hardware inicialmente***, portanto, você pode desenvolver e implantar aplicativos com mais rapidez.

Você ***pode usar o Amazon EC2 para executar o número de servidores virtuais que precisar***, configurar a segurança e a rede, e gerenciar o armazenamento.

O Amazon EC2 também permite a expansão ou a redução para gerenciar as alterações de requisitos ou picos de popularidade, reduzindo, assim, a sua necessidade de prever o tráfego do servidor.

AWS – EC2



<https://aws.amazon.com/pt/ec2/instance-types/>

AWS – EC2



<https://aws.amazon.com/pt/ec2/instance-types/>

A1	T4g	T3	T3a	T2	M6g	M5	M5a	M5n	M4	
----	-----	----	-----	----	-----	----	-----	-----	----	--

As instâncias M4 oferecem recursos equilibrados de computação, memória e rede e são uma boa opção para muitos aplicativos.

Recursos:

- Processadores Intel Xeon® E5-2686 v4 (Broadwell) de 2,3 GHz ou processadores Intel Xeon® E5-2676 v3 (Haswell) de 2,4 GHz
- Otimizado para EBS por padrão, sem custo adicional
- Suporte a redes avançadas
- Equilíbrio entre recursos de computação, memória e rede

Instância	vCPU*	Mem (GiB)	Armazenamento	Largura de banda dedicada do EBS (Mbps)	Performance de rede
m4.large	2	8	Somente EBS	450	Moderada
m4.xlarge	4	16	Somente EBS	750	Alta
m4.2xlarge	8	32	Somente EBS	1.000	Alta
m4.4xlarge	16	64	Somente EBS	2.000	Alta
m4.10xlarge	40	160	Somente EBS	4.000	10 Gigabit
m4.16xlarge	64	256	Somente EBS	10.000	25 Gigabit

AWS – EC2

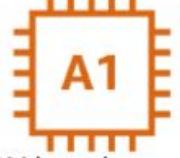
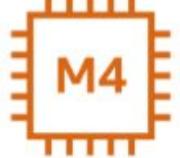
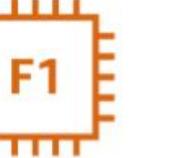


<https://aws.amazon.com/pt/ec2/instance-types/>

Instance	EC2 Compute Units	Memory (GiB)	Storage (GB)	Price per Hour
t2.small	variable	2.0	EBS	\$0.023
m5a.large	4	8.0	EBS	\$0.086
m5a.xlarge	8	16.0	EBS	\$0.172
c5.large	9	4.0	EBS	\$0.085
c5.xlarge	17	8.0	EBS	\$0.170

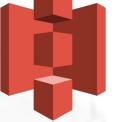
EC2 - Instâncias

IGTI

General Purpose	Compute Optimised	Memory Optimised	Accelerated Computing	Storage Optimised
 ARM based core and custom silicon	 Compute - CPU intensive apps and DBs	 RAM - Memory intensive apps and DB's	 Processing optimised-Machine Learning	 High Disk Throughput - Big data clusters
 Tiny - Web servers and small DBs		 Xtreme RAM - For SAP/Spark	 Graphics Intensive - Video and streaming	 IOPS - NoSQL DBs
 Main - App servers and general purpose		 High Compute and High Memory - Gaming	 Field Programmable - Hardware acceleration	 Dense Storage - Data Warehousing

Storage

IGTI

Amazon S3	Amazon EFS	Amazon Glacier	AWS Storage Gateway	AWS Snowball*	Amazon EBS
					
bucket	file system	archive	cached volume	import/export	snapshot
					volume
object			virtual tape library		

AWS – S3



O Amazon ***Simple Storage Service*** é armazenamento para a Internet. Ele foi projetado para facilitar a computação de escala na web para os desenvolvedores.

O Amazon S3 tem uma interface simples de serviços da web que você pode usar para armazenar e recuperar qualquer quantidade de dados, a qualquer momento, em qualquer lugar da web.

Ela concede acesso a todos os desenvolvedores para a mesma infraestrutura altamente dimensionável, confiável, segura, rápida e econômica que a Amazon utiliza para rodar a sua própria rede global de sites da web.

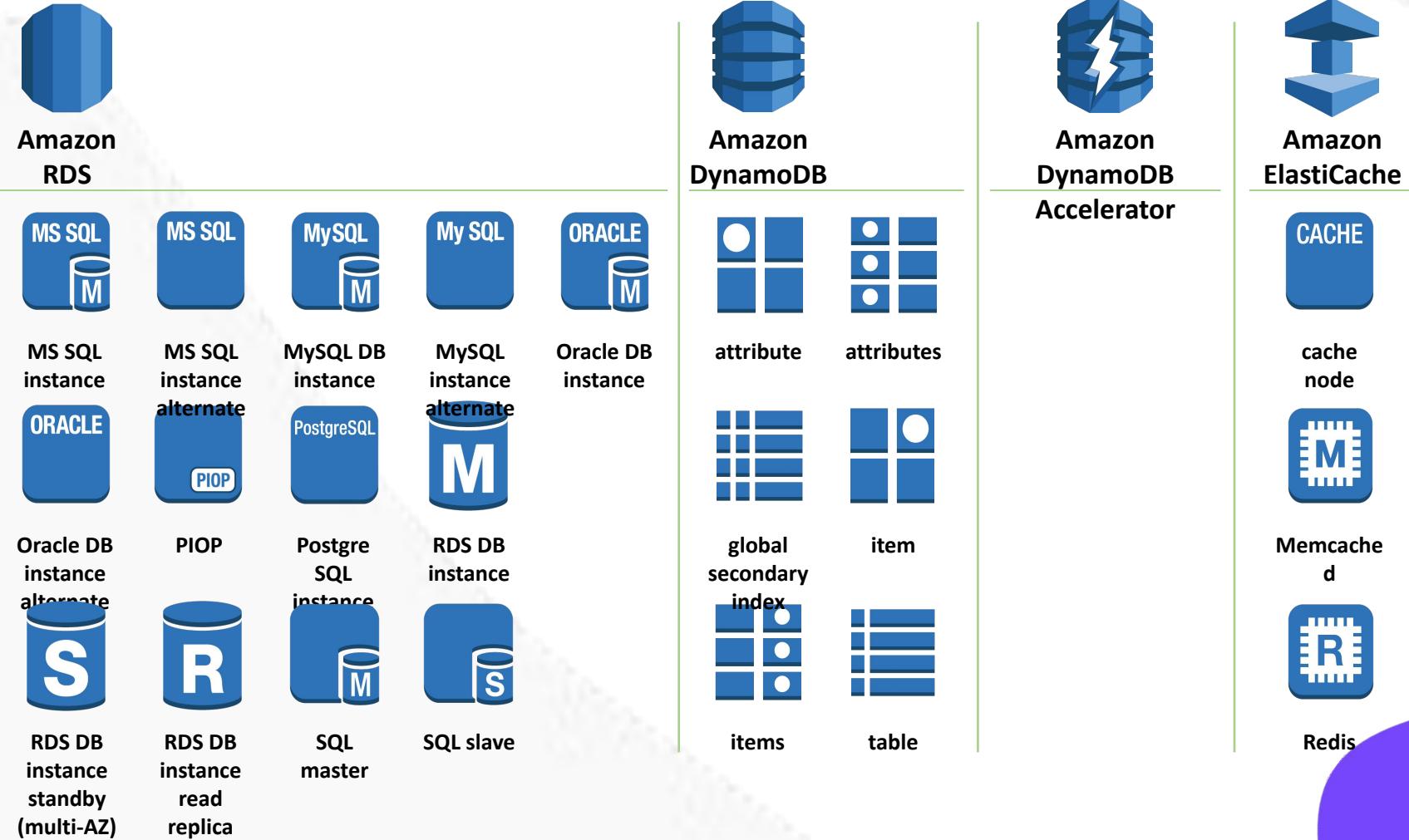
O serviço visa maximizar os benefícios de escala e poder passar esses benefícios para os desenvolvedores.

AWS – S3



	S3 Standard	S3 Intelligent-Tiering*	S3 Standard-IA	S3 One Zone-IA†	S3 Glacier	S3 Glacier Deep Archive**
Projetado para resiliência	99,999999999% (11 9s)	99,999999999% (11 9's)				
Projetado para disponibilidade	99,99%	99,9%	99,9%	99,5%	99,99%	99,99%
Acordo de nível de serviço de disponibilidade	99,9%	99%	99%	99%	99,9%	99,9%
Zonas de disponibilidade	≥3	≥3	≥3	1	≥3	≥3
Cobrança mínima de capacidade por objeto	N/D	N/D	128 KB	128 KB	40 KB	40 KB
Cobrança mínima de duração de armazenamento	N/D	30 dias	30 dias	30 dias	90 dias	180 dias
Taxa de recuperação	N/D	N/D	por GB recuperado	por GB recuperado	por GB recuperado	por GB recuperado
Latência de primeiro byte	milissegundos	milissegundos	milissegundos	milissegundos	selecione minutos ou horas	selecione horas
Tipo de armazenamento	Objeto	Objeto	Objeto	Objeto	Objeto	Objeto
Transições de ciclo de vida	Sim	Sim	Sim	Sim	Sim	Sim

Database



AWS – RDS



O Amazon *Relational Database Service* (Amazon RDS) *facilita a configuração, a operação e a escalabilidade de bancos de dados relacionais na nuvem.*

O serviço *oferece capacidade econômica e redimensionável e automatiza tarefas demoradas de administração, como provisionamento de hardware, configuração de bancos de dados, aplicação de patches e backups.*

Dessa forma, você pode se concentrar na performance rápida, alta disponibilidade, segurança e conformidade que os aplicativos precisam.

AWS – RDS



Mecanismos de banco de dados do Amazon RDS



AWS

IGTI

COMPUTE	
Free Tier	12 MONTHS FREE
Amazon EC2 750 Hours per month	
Resizable compute capacity in the Cloud.	

STORAGE	
Free Tier	12 MONTHS FREE
Amazon S3 5 GB of standard storage	
Secure, durable, and scalable object storage infrastructure.	

DATABASE	
Free Tier	12 MONTHS FREE
Amazon RDS 750 Hours per month of db.t2.micro database usage (applicable DB engines)	
Managed Relational Database Service for MySQL, PostgreSQL, MariaDB, Oracle BYOL, or SQL Server.	

DATABASE	
Free Tier	ALWAYS FREE
Amazon DynamoDB 25 GB of storage	
Fast and flexible NoSQL database with seamless scalability.	

MACHINE LEARNING	
Free Tier	FREE TRIAL
Amazon SageMaker 250 Hours per month of t2.medium notebook usage for the first two months	
Fully managed platform to build, train, and deploy machine learning models.	

COMPUTE	
Free Tier	ALWAYS FREE
AWS Lambda 1 Million free requests per month	
Compute service that runs your code in response to events and automatically manages the compute resources.	

Conclusão



- ✓ AWS
- ✓ Principais Serviços

Próxima Aula



01. •

Introdução ao Google Cloud

02. •

03. •

04. •

Armazenamento de Dados

Capítulo 04 – Aula 04.03 - Introdução ao Google Cloud

PROF.: RICARDO BRITO ALVES

Nesta aula



- Google Cloud
- Principais Serviços

Google Cloud Platform



Google Cloud *Platform* é uma suíte de computação em nuvem oferecida pelo Google, funcionando na mesma infraestrutura que a empresa usa para seus produtos dirigidos aos usuários, dentre eles o *Buscador Google* e o *Youtube*.

Juntamente com um conjunto de ferramentas de gerenciamento modulares, fornecem uma série de serviços *incluindo, computação, armazenamento de dados, análise de dados e aprendizagem de máquina.*

Compute Engine



Criação de máquinas virtuais

O Google Compute Engine é o componente ***Infraestrutura como serviço do Google Cloud Platform***, construído sobre a infraestrutura global que executa o mecanismo de pesquisa do Google, Gmail, YouTube e outros serviços.

O Google Compute Engine ***permite que os usuários iniciem máquinas virtuais sob demanda***. Uma máquina virtual é um computador emulado em outra máquina. Nesse caso, o servidor em nuvem executa vários sistemas operacionais em diversas outras máquinas, em vez de cada computador ter um sistema operacional próprio.

Compute Engine



O Google Compute Engine *oferece máquinas virtuais em execução nos centros de dados inovadores do Google e na rede mundial de fibra*. O suporte a ferramentas e fluxo de trabalho do Compute Engine permite dimensionar de instâncias únicas para computação em nuvem balanceada de carga global. *As VMs do Compute Engine são inicializadas rapidamente, vêm com opções de disco persistentes e locais de alto desempenho, oferecem desempenho consistente.*

- Instâncias.
- Configurações.
- Deploy de aplicação e Jobs.
- Alertas automáticos.

Storage & Databases



Amplo armazenamento de dados em nuvem.

A ideia de Plataforma como um Serviço (PaaS) já é bem difundida em muitos setores, especialmente no que diz respeito à gestão e ao armazenamento de dados.

É similar ao Google Drive, porém, em escala bem maior, o que permite a você armazenar e organizar todos os arquivos da empresa e acessá-los a partir de qualquer máquina com permissão de acesso. É ótimo para evitar a perda de documentos e para minimizar o uso do espaço físico em seu negócio.

Storage & Databases



Armazenamento de produtos escaláveis, resilientes e de alto desempenho. Bancos de dados para suas aplicações.

- Armazenamento de arquivos.
- Gerenciamento de buckets.
- Bucket Locations.
- Multi Regional.
- Regional.
- NearLine.
- CodeLine.
- Linha de comando, API, Console.

App Engine



Plataforma para criação de aplicativos Web escaláveis e backends para dispositivos móveis. O App Engine fornece serviços integrados e APIs, como datastores NoSQL, memcache e uma API de autenticação de usuário, comum à maioria dos aplicativos.

- Construir aplicações de desenvolvimentos escaláveis.
- Instâncias automáticas.
- Integrações.
- Deploy automatizado.

Big Data



Quando precisamos tomar uma decisão importante para a empresa, é importante ter dados que deem suporte a essa escolha. Ainda mais, atualmente, em um mercado cada vez mais complexo e volátil.

Totalmente gerenciado data warehousing, lote e processamento de fluxo, exploração de dados, Hadoop / Spark, e mensagens confiáveis.

- Análise de dados em bancos NoSQL.
- Construir bancos via: linha de comando, console e API's.
- Importar dados: csv, txt, integrações.

Machine Learning



Serviços de ML rápidos, escaláveis e fáceis de usar. Use modelos pré-treinados ou treine modelos personalizados em seus dados.

- Máquina de conhecimento, como funciona.
- Exemplo Case Google: E-mail e Spam.
- Google Cloud Vision API.
- Google Translate API.
- Google Speech API.

Google Cloud Platform



Google Cloud Pricing Calculator

Prices are up to date. Last update: 16-November-2020

COMPUTE ENGINE APP ENGINE KUBERNETES ENGINE CLOUD RUN VMWARE ENGINE CLOUD STORAGE NETWORKING EGRESS CLOUD LOAD BALANCING INTE & CL

Estimate

Search for a product you are interested in.

Instances

Number of instances *

What are these instances for?

Operating System / Software

Free: Debian, CentOS, CoreOS, Ubuntu, or other User Provided OS

Machine Class

Regular

Machine Family

General purpose

Google Cloud vs AWS



O Google Cloud e o AWS são bem semelhantes. Para fazermos uma comparação seria necessário observar as categorias abaixo:

- Compute.
- Armazenamento em Blocos.
- Rede.
- Faturamento e Preços.
- Suporte e tempo de atividade.
- Segurança.

Google Cloud vs AWS



Tipo de Máquina/Instância	Google Compute Engine	AWS EC2
Compartilhado	f1-micro g1-small	t2.nano – t2.2xlarge
Padrão	n1-standard-1 – n1-standard-96 (beta)	m3.medium – m3.2xlarge m4.large – m4.16xlarge
Alta memória	n1-higmem-2 – n1-higmem-96 (beta)	r3.large – r3.8xlarge r4.large – r4.16xlarge x1.16xlarge – x1e.32xlarge
Alto CPU	n1-highcpu-2 – n1-highcpu-96 (beta)	c3.large – c3.8xlarge c4.large – c4.8xlarge
GPU	You can add GPUs to machine types	g2.2xlarge g2.8xlarge
Armazenamento SSD	n1-standard-1 – n1-standard-32 n1-higmem-2 – n1-higmem-32 n1-highcpu-2 – n1-highcpu-32	i2.xlarge – i2.8xlarge
Armazenamento denso	N/A	d2.xlarge – d2.8xlarge

Google Cloud vs AWS

IGTI

Armazenamento em bloco	Google Cloud Platform	AWS
Serviço	SSD	SSD IOPS geral e provisionado
Tamanhos	1 GB até 64 TB	1 GB até 16 TB IOPS provisionados de 4 GB a 16 TB
IOPS máximos por volume	40,000 ler, 30,000 escrever	10.000 (20.000 para IOPS provisionados) IOPS máximo de 75.000 / instância
Rendimento Máximo por Volume (MB/s)	800 ler, 400 escrever	160 (320 para IOPS provisionados)
Replicação	Redundância incorporada	RAID-1
Redundância de snapshots	Múltiplas localizações	Múltiplas localizações
Encriptação	SSE 256-bit AES	SSE 256-bit AES
Encriptação	SSE 256-bit AES	SSE 256-bit AES
Preço Magnético (por GB / mês)	\$0.040 (disco padrão)	\$0.045
Preço de SSD (por GB/mês)	\$0.170	\$0.10
Preços de SSD PIOPS (por GB/mês)	N/A	\$0.125

Conclusão



- ✓ Google Cloud
- ✓ Principais Serviços

Próxima Aula



01. •

Introdução a Microsoft Azure

02. •

03. •

04. •

Armazenamento de Dados

Capítulo 04 – Aula 04.04 - Introdução a Microsoft Azure

PROF.: RICARDO BRITO ALVES

Nesta aula



- Microsoft Azure
- Principais Serviços

Microsoft Azure



O Microsoft Azure é uma plataforma destinada à execução de aplicativos e serviços, baseada nos conceitos da computação em nuvem.

A apresentação do serviço foi feita no dia 27 de outubro de 2008 durante a Professional Developers Conference, em Los Angeles e lançado em 1 de Fevereiro de 2010 como Windows Azure, para então ser renomeado como Microsoft Azure em 25 de Março de 2014.

Funcionamento

Sua computação em nuvem é definida como uma combinação de software como serviço (SaaS) com computação em grid.

A computação em grid dá o poder de computação e alta escalabilidade oferecida para as aplicações, através de milhares de máquinas (hardware) disponíveis em centros de processamento de dados de última geração. De software como serviço se tem a capacidade de contratar um serviço e pagar somente pelo uso, permitindo a redução de custos operacionais, com uma configuração de infraestrutura realmente mais aderente às necessidades.

Recursos

Além dos recursos de computação, armazenamento e administração oferecidos pelo Microsoft Azure, a plataforma também disponibiliza uma série de serviços para a construção de aplicações distribuídas, além da total integração com a solução on-premise (local) baseada em plataforma .NET.

Entre os principais serviços da plataforma Windows Azure há o SQL Azure Database, Azure AppFabric Platform e uma API de gerenciamento e monitoramento para aplicações colocadas na nuvem.

Microsoft Azure x AWS



Alguns pontos em comum entre a AWS e Azure:

- Autonomia e provisionamento
- Escalabilidade instantânea
- Segurança
- Conformidade
- Gerenciamento de identidade

As formas de cobrança da AWS podem ser:

On-demand: O cálculo é feito em cima de horas ou segundos utilizados (no mínimo 60 segundos) e somente as instâncias EC2 que forem utilizadas;

Instâncias reservadas (RIs): O preço por hora é fixo, independentemente do uso, e existe um prazo pré-determinado de contratação. Essa forma de pagamento tem o benefício de obter desconto, uma vez que o cliente tem o compromisso de um a três anos;

Instâncias spot: Por serem instâncias extras, ou seja, instâncias de capacidade extra na Nuvem AWS, o preço é muito mais atrativo. Por outro lado, se o EC2 precisar de capacidade, o cliente que utiliza Instância Spot será notificado 2 minutos antes que suas instâncias serão interrompidas.

Microsoft Azure



As formas de cobrança da Azure podem ser:

On-demand: Seus custos são realizados em cima dos minutos utilizados.

Neste modelo, não é necessário compromisso de tempo mínimo de contratação. Como o pagamento é feito de acordo com a utilização, é possível aumentar e diminuir recursos sem limite.

Contrato pré-definido: Como um determinado tempo de utilização é acordado, o custo é reduzido.

Acordo empresarial: Nessa modalidade, o pagamento é realizado antecipadamente, por esse motivo, há benefícios e desconto. O uso adicional é pago de forma separada, mas com desconto nas taxas.

Conclusão



- ✓ Microsoft Azure
- ✓ Principais Serviços

Próxima Aula



01. ••

Arquitetura Microserviços

02. ••

03. ••

04. ••

Armazenamento de Dados

Capítulo 04 – Aula 04.05 – Arquitetura Microserviços

PROF.: RICARDO BRITO ALVES

Nesta aula

- Arquitetura Monolítica
- Arquitetura Microserviços
- Containers
- Orquestradores de Container

Arquiteturas Monolíticas



Com as arquiteturas monolíticas, todos os processos são altamente acoplados e executam como um único serviço.

Isso significa que se um processo do aplicativo apresentar um pico de demanda, toda a arquitetura deverá ser escalada. A complexidade da adição ou do aprimoramento de recursos de aplicativos monolíticos aumenta com o crescimento da base de código. Essa complexidade limita a experimentação e dificulta a implementação de novas ideias.

As arquiteturas monolíticas aumentam o risco de disponibilidade de aplicativos, pois muitos processos dependentes e altamente acoplados aumentam o impacto da falha de um único processo.

Arquitetura de Microserviços

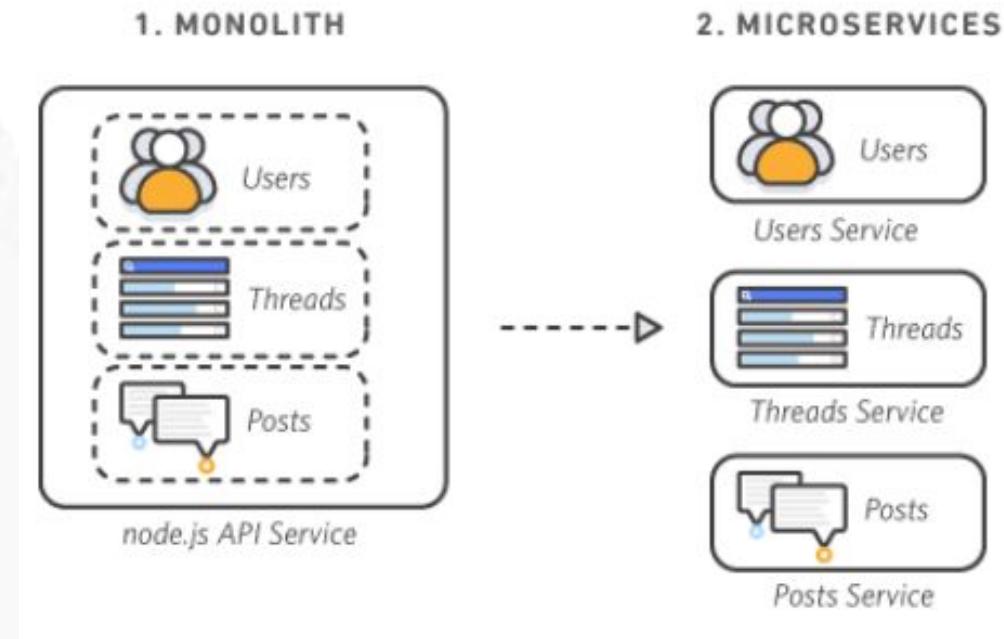


Com uma arquitetura de microserviços, um aplicativo é criado como componentes independentes que executam cada processo do aplicativo como um serviço.

Esses serviços se comunicam por meio de uma interface bem definida usando APIs leves. Os serviços são criados para recursos empresariais e cada serviço realiza uma única função. Como são executados de forma independente, cada serviço pode ser atualizado, implantado e escalado para atender a demanda de funções específicas de um aplicativo.

Arquitetura de Microserviços

IGTI



Arquitetura de Microserviços



Escalabilidade flexível

Os microserviços permitem que cada serviço seja escalado de forma independente para atender à demanda do recurso de aplicativo oferecido por esse serviço. Isso permite que as equipes dimensionem corretamente as necessidades de infraestrutura, meçam com precisão o custo de um recurso e mantenham a disponibilidade quando um serviço experimenta um pico de demanda.



Container

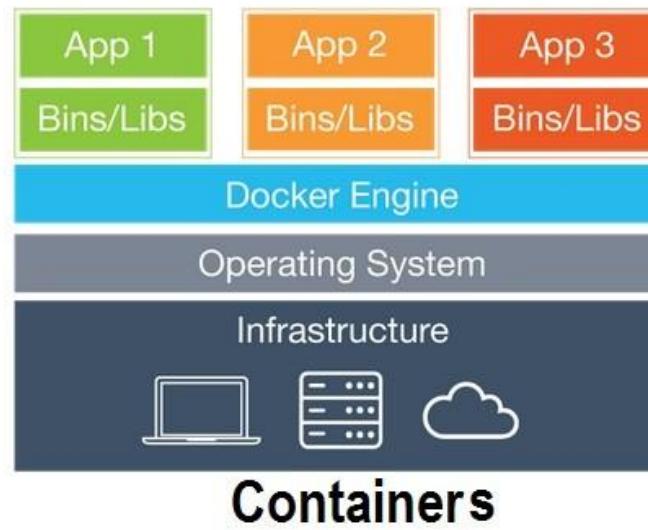
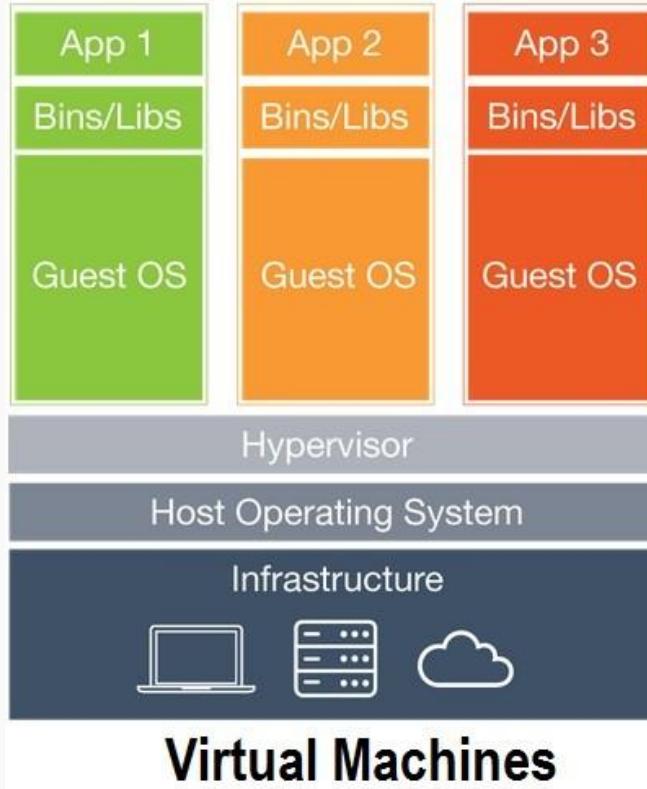


Em vez de usar um sistema operacional para cada estrutura, como na virtualização, os Containers são blocos de espaços divididos pelo Docker em um servidor, o que possibilita a implementação de estruturas de Microserviços que compartilham o mesmo sistema operacional. Porém, de forma limitada (conforme a demanda por capacidade).

O fato de os Containers não terem seus próprios sistemas operacionais, permite que eles consumam menos recursos e, com isso, sejam mais leves.

Container

IGTI



Ferramentas de Orquestração de Containers



As ferramentas de orquestração de containers são aplicações em nuvem que permitem fazer o gerenciamento de múltiplos contêineres.

Seus principais objetivos são:

- Cuidar do ciclo de vida dos containers de forma autônoma, subindo e distribuindo, conforme nossas especificações ou demandas;
- Gerenciar volumes e rede, que podem ser local ou no cloud provider de sua preferência.

Orquestradores de Containers



O Kubernetes, ECS e o Docker são as principais plataformas de gerenciamento de contêineres.

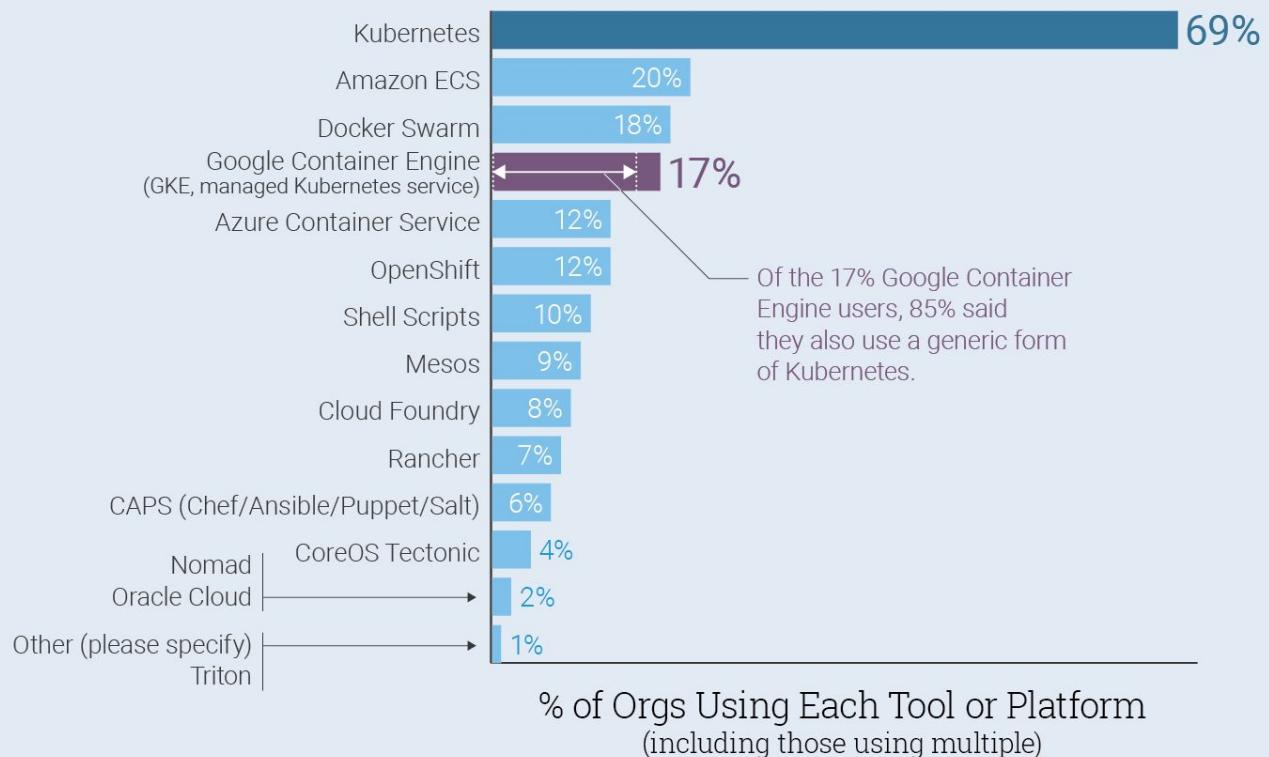
Dessa forma, essa é uma ferramenta para viabilizar a utilização de Containers e Microserviços em servidores com mais facilidade, pois permite empacotar os aplicativos para que possam ser movimentados facilmente.

O Docker permite, por exemplo, que uma biblioteca possa ser instalada em diferentes Containers sem que haja qualquer interdependência entre eles. Essa característica tem o objetivo de facilitar o gerenciamento de códigos e aplicativos.

Orquestradores de Containers

IGTI

Kubernetes Manages Containers at 69% of Organizations Surveyed



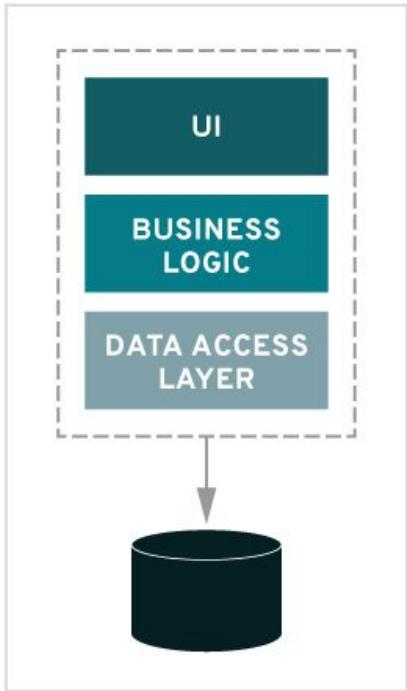
Source: The New Stack Analysis of Cloud Native Computing Foundation survey conducted in Fall 2017.
Q. Your organization manages containers with... (check all that apply)? n=763.

THE NEW STACK

Arquitetura

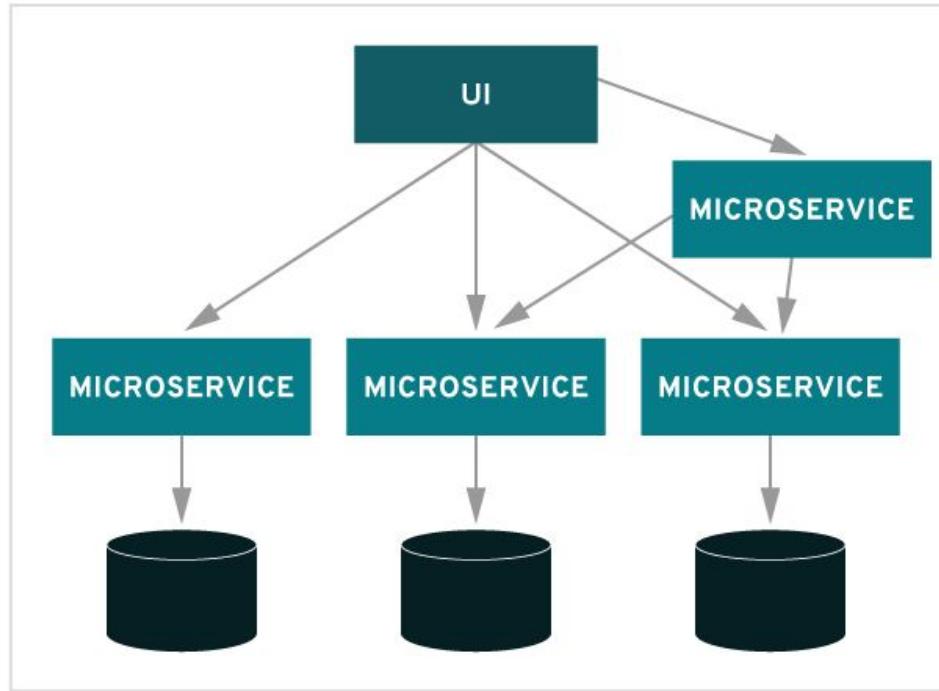
IGTI

MONOLITHIC



VS.

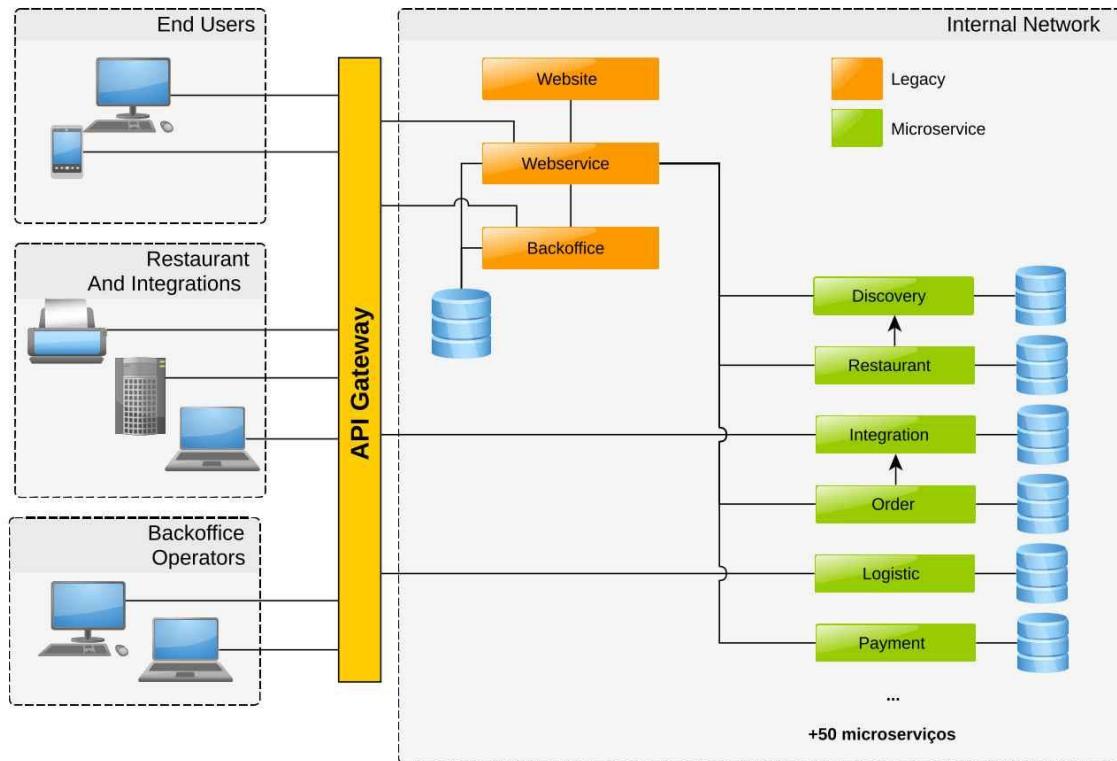
MICROSERVICES



Arquitetura

IGTI

Arquitetura de micro-serviços



Conclusão



- ✓ Arquitetura Monolítica
- ✓ Arquitetura Microserviços
- ✓ Containers
- ✓ Orquestradores de Container

Próxima Aula



01. ••

03. ••

Sistema de Arquivos Distribuídos

02. ••

04. ••

Armazenamento de Dados

Capítulo 04 – Aula 04.06 – Sistemas de Arquivo Distribuídos

PROF.: RICARDO BRITO ALVES

Nesta aula

- Sistemas de Arquivo
- Sistemas de Arquivo Distribuído
- Apache Hadoop
- HDFS

Sistemas de Arquivos



- Foram originalmente desenvolvidos como um *recurso do S.O* que fornece uma interface de programação conveniente para armazenamento em disco.
- *São responsáveis pela organização, armazenamento, recuperação, atribuição de nomes, compartilhamento e proteção de arquivos.*
- Projetados para *armazenar e gerenciar um grande número de arquivos*, com recursos para criação, atribuição de nomes e exclusão de arquivos.

Sistemas de Arquivos

- Diretório é um arquivo de tipo especial;
- Fornece um mapeamento dos nomes textuais para identificadores internos;
- Podem incluir nomes de outros diretórios.

Tamanho do Arquivo
Horário de Criação
Horário de Acesso (Leitura)
Horário de Modificação (Escrita)
Horário de Alteração de Atributo
Contagem de Referência
Proprietário
Tipo de Arquivo
Lista de Controle de Acesso

Sistemas de Arquivos Distribuídos (DFS ou SAD)



Um sistema de arquivos distribuídos *permite aos programas armazenarem e acessarem arquivos remotos exatamente como se fossem locais*, possibilitando que os usuários acessem arquivos a partir de qualquer computador em uma rede.” (COULOURIS, et. al, p. 284).

- **Objetivo:** permitir que os programas armazenem e acessem arquivos remotos exatamente como se fossem locais.
- **Permitem que vários processos** compartilhem dados por longos períodos, de modo seguro e confiável.
- **O desempenho e segurança** no acesso aos arquivos armazenados em um servidor devem ser compatíveis aos arquivos armazenados em discos locais.

Requisitos de um Sistemas de Arquivos Distribuídos



- Transparência.
- Atualização concorrente de arquivos.
- Replicação de arquivos.
- Heterogeneidade.
- Tolerância a falha.
- Consistência.
- Segurança.
- Eficiência.

Arquitetura do SAD

- Modelo abstrato de arquitetura que serve para o ***Network File System (NFS)*** e ***Andrew File System (AFS)***.
- Divisão de responsabilidades entre três módulos.
 - Cliente.
 - Serviço de arquivos planos.
 - Serviço de diretórios.
- Design aberto
 - Diferentes módulos cliente podem ser utilizados para implementar diferentes interfaces.
 - Simulação de operações de arquivos de diferentes S.O.
 - Otimização de performance para diferentes configurações de hardware de clientes e servidores.



Sistemas de Arquivos Distribuídos (DFS ou SAD)



Funções de um Sistema de Arquivos Distribuído:

- ***Armazenar e compartilhar programas e dados***
 - Funções idênticas às de um sistema centralizado (local).
- ***Ênfase na disponibilidade, confiabilidade e segurança.***
- ***Desempenho***
 - Questão importante porque acessos remotos podem ser significativamente mais lentos que os locais.
 - Não se pretende em geral que o SAD seja mais rápido que um SA local, mas sim que a degradação seja aceitável.

Sistemas de Arquivos Distribuídos (DFS ou SAD)



Sistemas de arquivo distribuídos *devem ser vistos pelos clientes como um sistema de arquivo local.*

A *transparência* é muito importante para seu bom funcionamento.

É necessário *um bom controle de concorrência* no acesso.

Cache é importante.

Apache Hadoop



Hadoop é uma *plataforma de software de código aberto para o armazenamento e processamento distribuído de grandes conjuntos de dados, utilizando clusters de computadores com hardware commodity.*

Os serviços do Hadoop fornecem armazenamento , processamento, acesso, governança, segurança e operações de Dados.

Benefícios do Apache Hadoop



Algumas das razões para se usar Hadoop é a sua “capacidade de armazenar, gerenciar e analisar grandes quantidades de dados estruturados e não estruturados de forma rápida, confiável, flexível e de baixo custo.

- **Escalabilidade e desempenho** – distribuídos tratamento de dados local para cada nó em um cluster Hadoop permite armazenar, gerenciar, processar e analisar dados em escala petabyte.
- **Confiabilidade** – clusters de computação de grande porte são propensos a falhas de nós individuais no cluster. Hadoop é fundamentalmente resistente – quando um nó falha de processamento é redirecionado para os nós restantes no cluster e os dados são automaticamente re-replicado em preparação para falhas de nó futuras.

Benefícios do Apache Hadoop



- **Flexibilidade** – ao contrário de sistemas de gerenciamento de banco de dados relacionais tradicionais, você não tem que esquemas estruturados criados antes de armazenar dados. *Você pode armazenar dados em qualquer formato, incluindo formatos semi-estruturados ou não estruturados*, e em seguida, analisar e aplicar esquema para os dados quando ler.
- **Baixo custo** – ao contrário de software proprietário, *o Hadoop é open source* e é executado em hardware commodity de baixo custo.

HDFS



O HDFS (Hadoop Distributed File System) é um sistema de arquivos distribuído, *projeto para armazenar arquivos muito grandes*, com padrão de acesso aos dados streaming , utilizando clusters de servidores facilmente encontrados no mercado e de baixo ou médio custo.

Não deve ser utilizado para aplicações que precisem de acesso rápido a um determinado registro e sim para aplicações nas quais é necessário ler uma quantidade muito grande de dados.

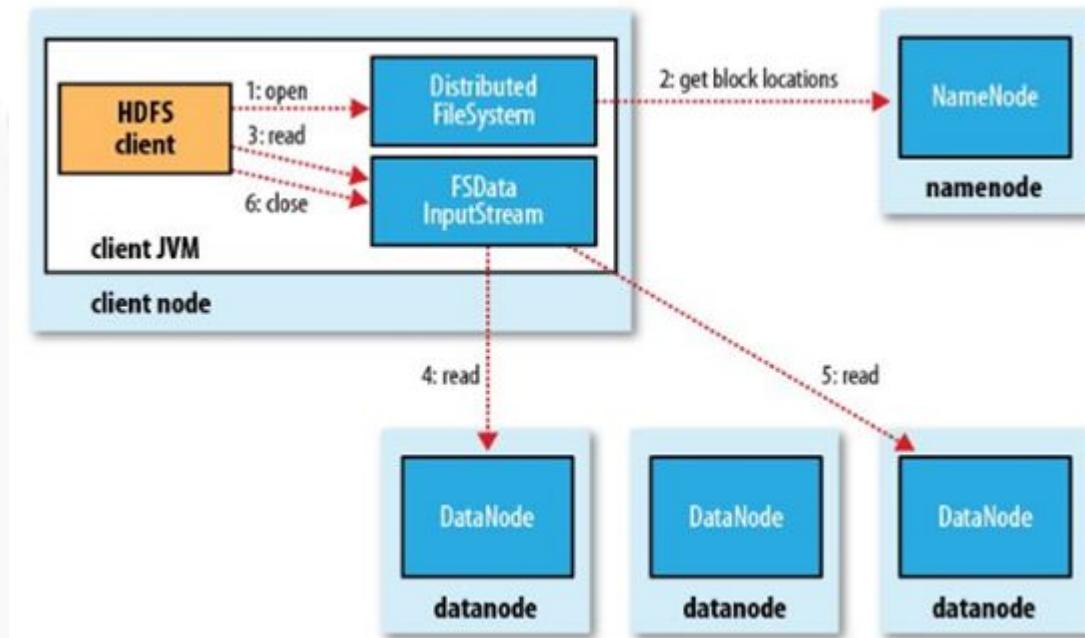
Outra questão que deve ser observada é que *não deve ser utilizado para ler muitos arquivos pequenos*, tendo em vista o overhead de memória envolvido.

Recursos do HDFS

- O HDFS tem 2 tipos de Nós : **Master** (ou Namenode) e **Worker** (ou Datanode).
 - O **Master** armazena informações da distribuição de arquivos e metadados.
 - Já o **Worker** armazena os dados propriamente ditos. Logo o Master precisa sempre estar disponível. Para garantir a disponibilidade podemos ter um backup (similar ao Cold Failover) ou termos um Master Secundário em um outro servidor. Nesta segunda opção, em caso de falha do primário, o secundário pode assumir o controle muito rapidamente.
- Tal como um sistema Unix, é possível utilizar o HDFS via linha de comando.

HDFS

IGTI



Conclusão



- ✓ Sistemas de Arquivo
- ✓ Sistemas de Arquivo Distribuído
- ✓ Apache Hadoop
- ✓ HDFS

Próxima Aula



01. ••

MongoDB na Nuvem

02. ••

03. ••

04. ••

Armazenamento de Dados

Capítulo 04 – Aula 04.07 – MongoDB na Nuvem

PROF.: RICARDO BRITO ALVES

Nesta aula

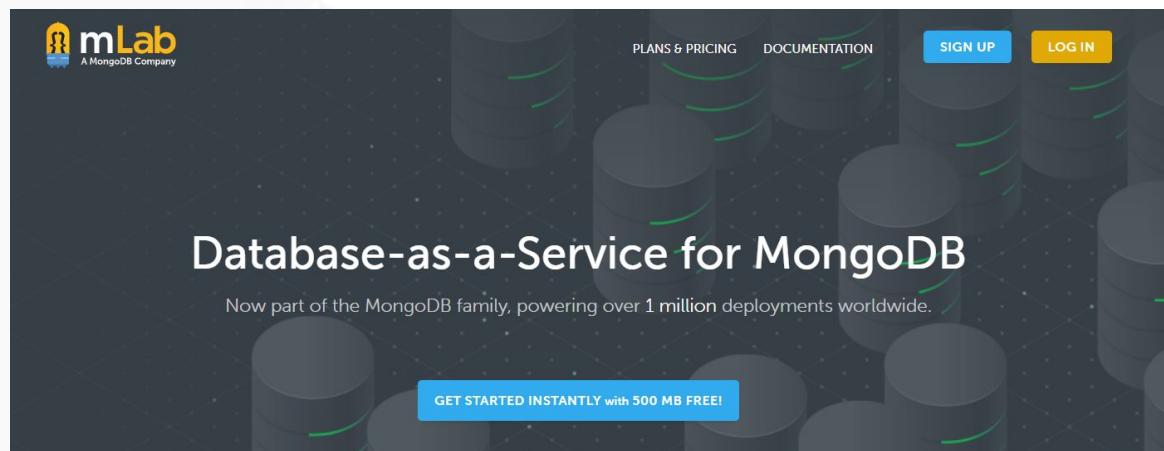
- Configurando mLab
- Conectando ao mLab

MongoDB Gratuito Hospedado na Nuvem



Conhecido um tempo atrás como MongoLab, o mLab é um serviço de banco de dados gerenciável que hospeda na nuvem um banco de dados MongoDB e é executado em provedores como a Amazon Web Services (AWS), Google Cloud e Microsoft Azure.

A parte mais interessante é que o serviço tem um plano gratuito que oferece 0,5 Gb para armazenamento de dados.



Criando uma Conta

Para que possamos utilizar o serviço, precisamos criar uma conta. O processo é bem simples e pode ser feito neste link (<https://mlab.com/>).

Nele você encontrará um formulário, basta preencher os dados e confirmar a conta no e-mail. É bem simples.

mLab is now part of the MongoDB family

If you're looking for a cloud-hosted MongoDB service similar to mLab, sign up for MongoDB Atlas, a fully-managed database-as-a-service available on AWS, Azure, and GCP

Create your account to start building your first cluster:

- Pick your preferred cloud provider: AWS, Azure, or Google
- Choose from over 60 cloud regions around the world
- Select your cluster tier and customize your storage
- Enable multi-region, workload isolation, and replication options for dedicated clusters
- Configure additional settings, including backup snapshots, sharded clusters, advanced security, and more

For more information on MongoDB Atlas pricing, features, and support, visit the [MongoDB Atlas page](#).

Try MongoDB Atlas

Used by millions of developers around the world.

Your Company (optional)

How are you using MongoDB? I'm learning MongoDB

Your Work Email

First Name Ricardo

Last Name Alves

Password

✓ 8 characters minimum

I agree to the [terms of service](#) and [privacy policy](#).

Get Started with 512 MB Free

Primeiro Acesso



Na próxima tela você terá duas escolhas a fazer: **o provedor e o plano**. Dos provedores, temos três possibilidades:

- Amazon Web Services (AWS)
- Google Cloud Platform
- Microsoft Azure

Cloud Provider & Region

AWS, N. Virginia (us-east-1) ▾

The screenshot shows a user interface for selecting a cloud provider and region. At the top, there are three buttons: AWS (highlighted with a green border), Google Cloud, and Azure. Below this, a dropdown menu is open, showing "AWS, N. Virginia (us-east-1)" as the selected option. The interface is divided into three main regions: ASIA, NORTH AMERICA, and EUROPE. Under ASIA, there are two options: Singapore (ap-southeast-1)★ and Mumbai (ap-south-1). Under NORTH AMERICA, there are three options: N. Virginia (us-east-1)★ (highlighted with a green border), Oregon (us-west-2)★, and Frankfurt (eu-central-1)★. Under EUROPE, there are two options: Ireland (eu-west-1)★ and Frankfurt (eu-central-1)★.

Region	Provider	Region ID	Status
ASIA	Singapore	(ap-southeast-1)	★ Recommended region
	Mumbai	(ap-south-1)	★
NORTH AMERICA	N. Virginia	(us-east-1)	★ Recommended region
	Oregon	(us-west-2)	★
	Frankfurt	(eu-central-1)	★
EUROPE	Ireland	(eu-west-1)	★
	Frankfurt	(eu-central-1)	★

Primeiro Acesso



Dos planos, também temos outras três opções:

Choose a path. Adjust anytime.

Available as a fully managed service across 60+ regions on AWS, Azure, and Google Cloud

Dedicated Multi-Region Clusters

For teams developing world-class applications that require multi-region resiliency or ultra-low latency.

- ✓ Includes all features from Shared and Dedicated Clusters
- ✓ Replicate data across multiple regions
- ✓ Globally distributed read and write operations
- ✓ Control data residency at the document level

Create a cluster

Starting at **\$0.13/hr***

*estimated cost \$98.55/month

Dedicated Clusters

For teams building applications that need advanced development and production-ready environments.

- ✓ Includes all features from Shared Clusters
- ✓ Auto-scaling
- ✓ Network isolation
- ✓ Realtime performance metrics

Create a cluster

Starting at **\$0.08/hr***

*estimated cost \$56.94/month

Shared Clusters

For teams learning MongoDB or developing small applications.

- ✓ Highly available auto-healing cluster
- ✓ End-to-end encryption
- ✓ Role-based access control

Create a cluster

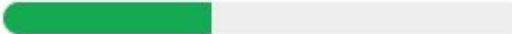
Starting at **FREE**

Guia



Connect to Atlas

Follow this checklist to get started.

40% 

- Build your first cluster
- Create your first database user
- Whitelist your IP address
- Load Sample Data (Optional)
- Connect to your cluster

No thanks

Criando Clusters



RICARDO'S ORG - 2020-11-29 > PROJECT 0

Clusters

SANDBOX

Cluster0

Version 4.2.10

[CONNECT](#) [METRICS](#) [COLLECTIONS](#) [...](#)

CLUSTER TIER

M0 Sandbox (General)

REGION

AWS / N. Virginia (us-east-1)

TYPE

Replica Set - 3 nodes

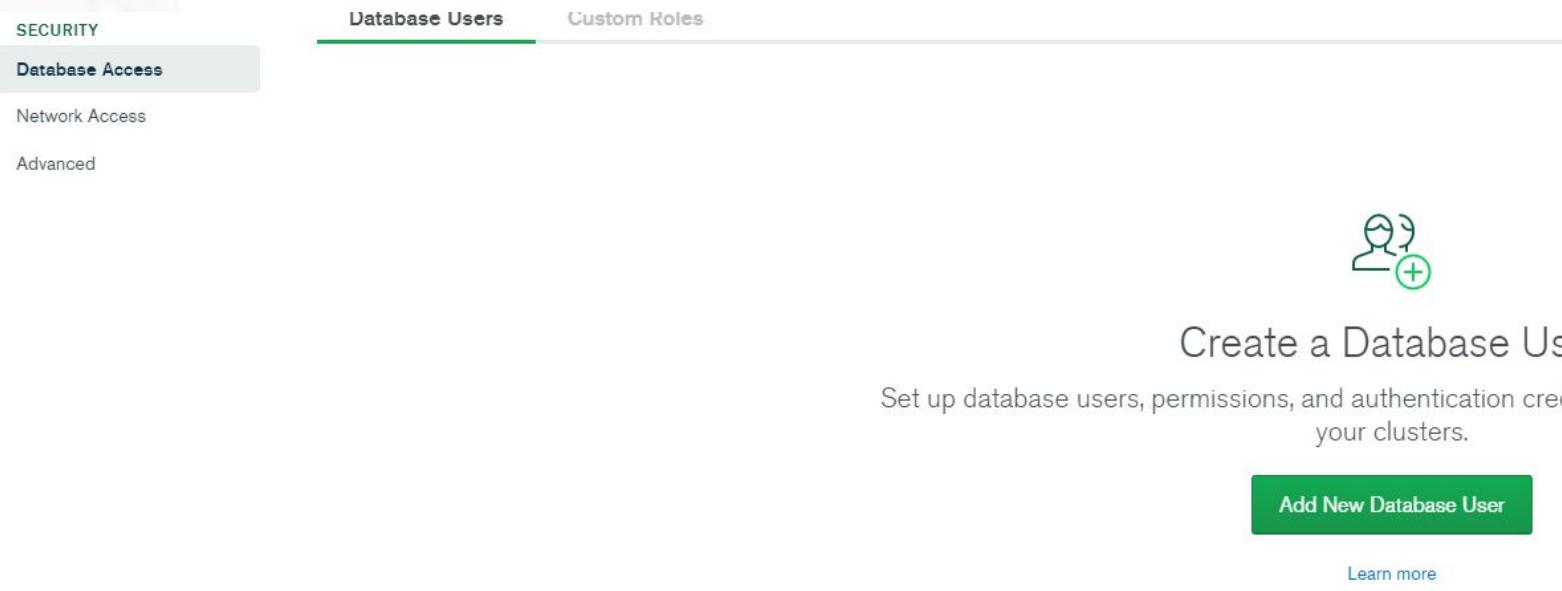
LINKED REALM APP

None Linked

Your cluster is being created

New clusters take between 1-3 minutes to provision.

Criando o Database User



The screenshot shows the AWS Database User creation interface. On the left, there's a sidebar with 'SECURITY' at the top, followed by 'Database Access' (which is highlighted in green), 'Network Access', and 'Advanced'. The main area has three tabs: 'Database Users' (highlighted in green), 'Custom Roles', and 'Custom Policies'. Below the tabs, there's a large icon of a user profile with a plus sign. The text 'Create a Database User' is centered, followed by the sub-instruction 'Set up database users, permissions, and authentication credentials for your clusters.' A green button labeled 'Add New Database User' is at the bottom, and a 'Learn more' link is just below it.

SECURITY

Database Access

Custom Roles

Database Access

Network Access

Advanced

Create a Database User

Add New Database User

Learn more

IP Access



Add IP Access List Entry

Atlas only allows client connections to a cluster from entries in the project's IP Access List. Each entry should either be a single IP address or a CIDR-notated range of addresses. [Learn more](#).

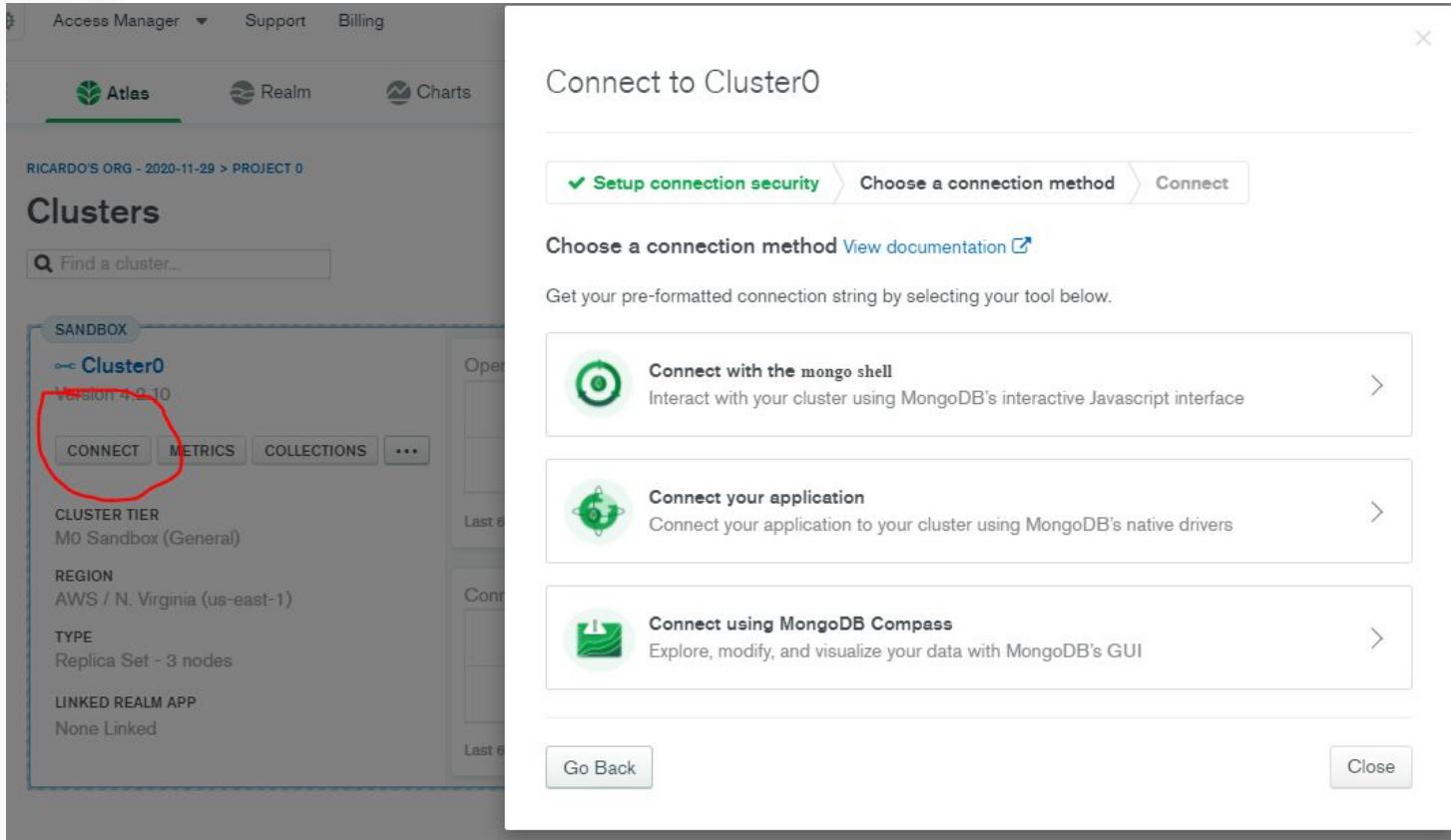
Access List Entry:

Comment:

This entry is temporary and will be deleted in

Conexão ao Cluster

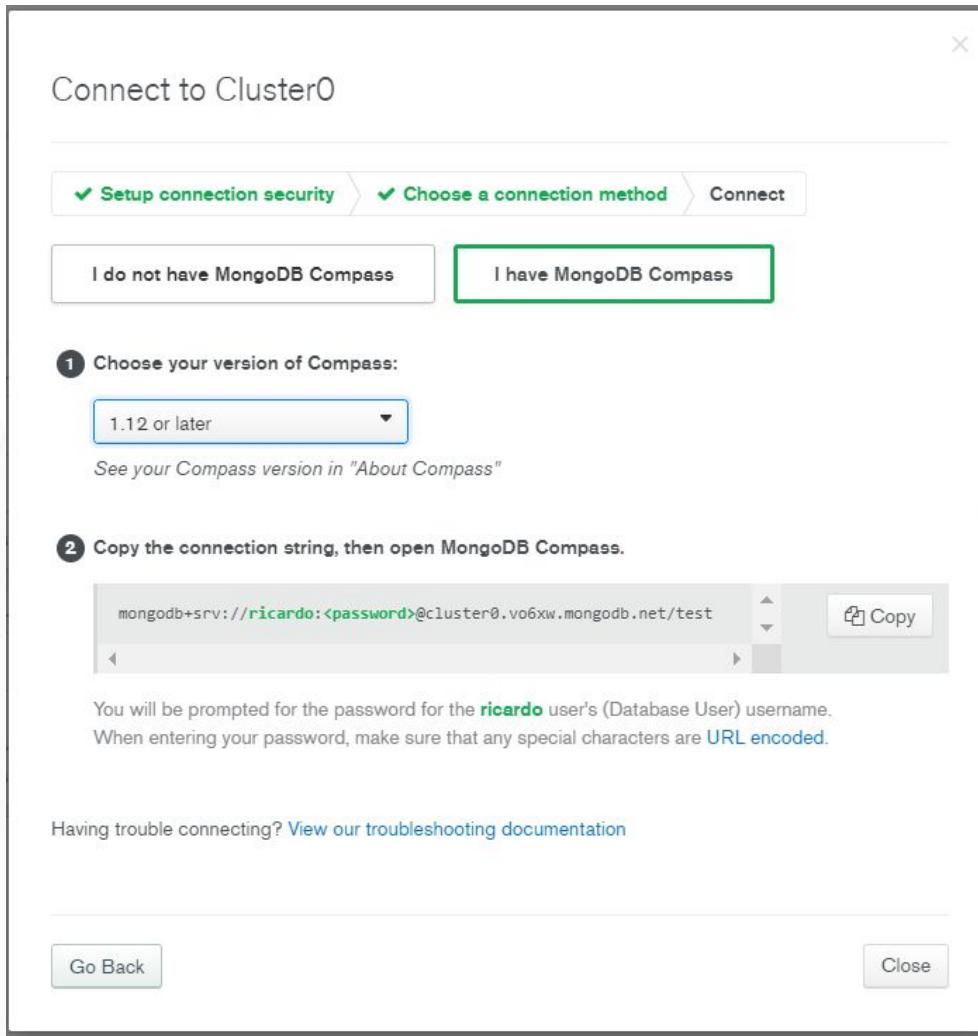
IGTI



The screenshot shows the MongoDB Atlas interface. On the left, there's a sidebar with 'Access Manager', 'Support', and 'Billing'. Below that is the 'Atlas' tab, which is underlined, followed by 'Realm' and 'Charts'. Under 'Clusters', it says 'RICARDO'S ORG - 2020-11-29 > PROJECT 0'. A search bar says 'Find a cluster...'. In the main area, there's a 'Sandbox' section with a 'Cluster0' entry. This entry has a red circle around the 'CONNECT' button. Below it are 'CLUSTER TIER' (M0 Sandbox (General)), 'REGION' (AWS / N. Virginia (us-east-1)), 'TYPE' (Replica Set - 3 nodes), and 'LINKED REALM APP' (None Linked). To the right of the 'Cluster0' entry is a 'Logs' section with 'Last 6 hours' and 'Conn' buttons. A modal window titled 'Connect to Cluster0' is open. It has a navigation bar with 'Setup connection security' (marked with a green checkmark), 'Choose a connection method' (marked with a blue arrow), and 'Connect'. The 'Choose a connection method' section contains three options: 'Connect with the mongo shell' (description: 'Interact with your cluster using MongoDB's interactive Javascript interface'), 'Connect your application' (description: 'Connect your application to your cluster using MongoDB's native drivers'), and 'Connect using MongoDB Compass' (description: 'Explore, modify, and visualize your data with MongoDB's GUI'). At the bottom of the modal are 'Go Back' and 'Close' buttons.

Conexão ao Cluster

IGTI



MongoDB Compass

IGTI

MongoDB Compass - cluster0.vo6xw.mongodb.net:27017

Connect View Help

Local

3 DBS 7 COLLECTIONS C

☆ FAVORITE

HOSTS

cluster0-shard-00-01.vo6x...
cluster0-shard-00-02.vo6x...
cluster0-shard-00-00.vo6x...

CLUSTER

Replica Set (atlas-13e3rw-...
3 Nodes

EDITION

MongoDB 4.2.10 Enterprise

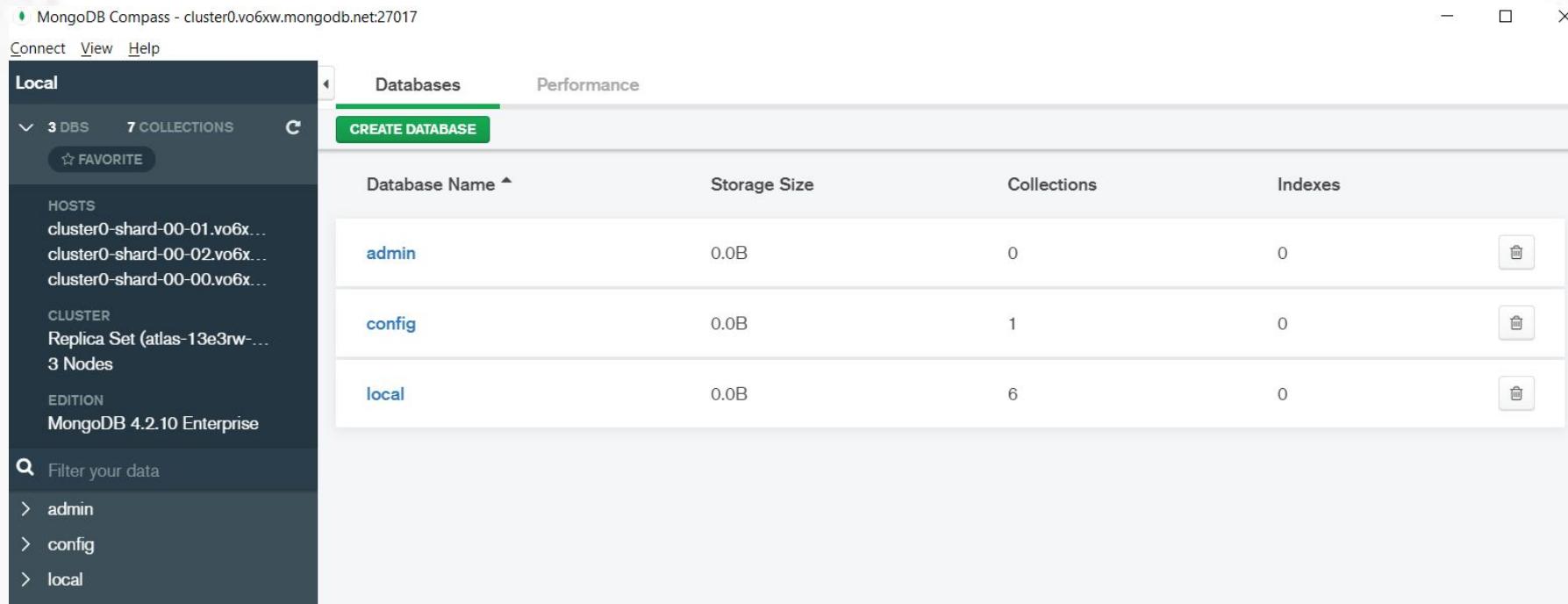
Filter your data

> admin
> config
> local

Databases Performance

CREATE DATABASE

Database Name	Storage Size	Collections	Indexes
admin	0.0B	0	0
config	0.0B	1	0
local	0.0B	6	0



Conclusão



MongoDB na nuvem:

- ✓ Configurando mLab
- ✓ Conectando ao mLab

Próxima Aula



01. ••

Data Warehouse e Data Lake

02. ••

03. ••

04. ••

Armazenamento de Dados

Capítulo 05 – Data Warehouse e Data Lake

PROF.: RICARDO BRITO ALVES

Armazenamento de Dados

Capítulo 05 – Aula 05.01 – Definição de BI, Data Warehouse e Data
Lake

PROF.: RICARDO BRITO ALVES

Nesta aula



- Definição de BI
- Data Warehouse

Business Intelligence



O termo Business Intelligence (BI), inteligência de negócios, **refere-se ao processo de coleta, organização, análise, compartilhamento e monitoramento de informações que oferecem suporte a gestão de negócios.**

É o conjunto de teorias, metodologias, processos, estruturas e tecnologias que transformam uma grande quantidade de dados brutos em informação útil para tomadas de decisões estratégicas.



Business Intelligence

- ✓ Descreve a capacidade da empresa ter acesso e explorar seus dados, desenvolvendo percepção e conhecimento, o que leva à melhora do processo de tomada de decisões.
- ✓ Sua infraestrutura tecnológica é composta de ***Data Warehouse ou Data Marts, data mining e ODS***, Benchmarking, além das ferramentas pertinentes.



O que é BI?



O que é BI?

IGTI

CONHECIMENTO

*... e, Sistemas de BI -
Business Intelligence*

INFORMAÇÃO

*Relatórios, gráficos e
cruzamentos*

DADOS

*Planilhas, sistemas
ERP, CRM e outros*

O que é BI?

- ✓ Business Intelligence (BI) é algo como obter as informações certas, para os tomadores de decisão certos, no momento certo.
- ✓ BI é uma plataforma que suporta relatórios, análises e tomada de decisão.
- ✓ **O BI proporciona:**
 - **Uma tomada de decisão baseada em fatos.**
 - **Visão única dos dados.**

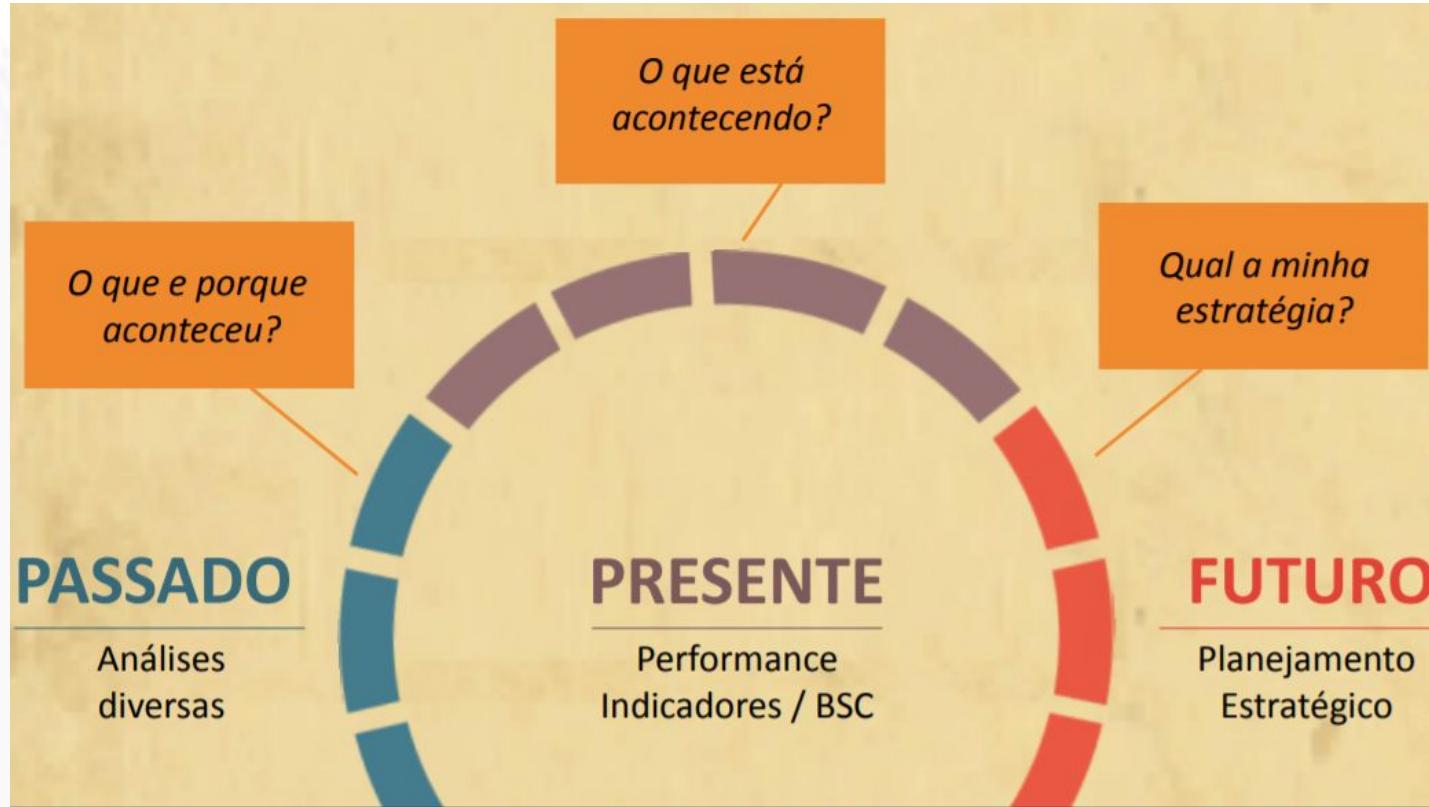
O que é BI?

O BI pode ser usado para adquirir

- ✓ Insights táticos para otimizar processos de negócios, identificando tendências, anomalias e comportamentos que requerem ação de gerenciamento.
- ✓ Visão estratégica para alinhar vários processos de negócios aos principais objetivos de negócios por meio de gerenciamento e análise de desempenho integrados.

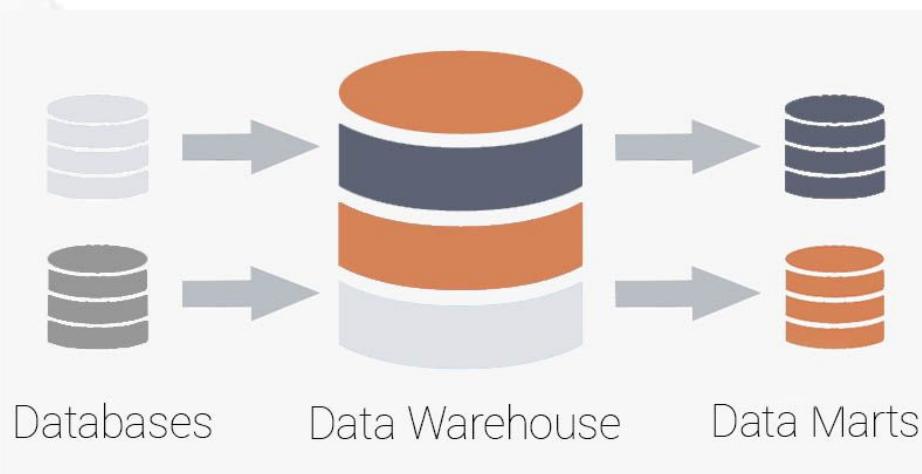
O que é BI?

IGTI



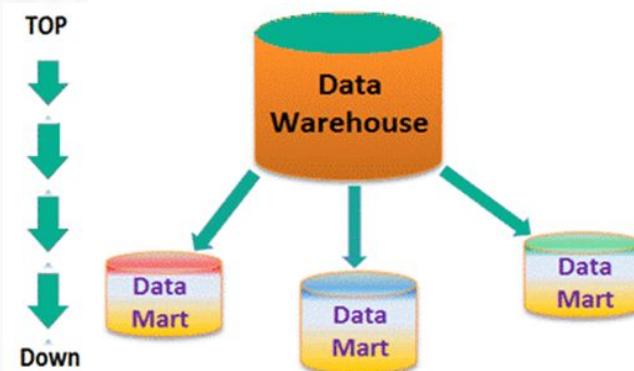
Data Warehouse (DW)

- ✓ Data Warehouse é um *depósito de dados digitais que serve para armazenar informações detalhadas relativamente a uma empresa, criando e organizando relatórios através de históricos* que são depois usados pela empresa para ajudar a tomar decisões importantes com base nos fatos apresentados.



Data Mart

- ✓ Um Data Mart é uma ***subdivisão ou subconjunto de um DW***. Os data marts são como pequenas fatias que armazenam subconjuntos de dados, normalmente organizados para um departamento ou um processo de negócio.
- ✓ Normalmente o Data Mart é ***direcionado para uma linha de negócios*** ou equipe, sendo que a sua informação costuma pertencer a um único departamento.



OLAP

Online Analytical Processing



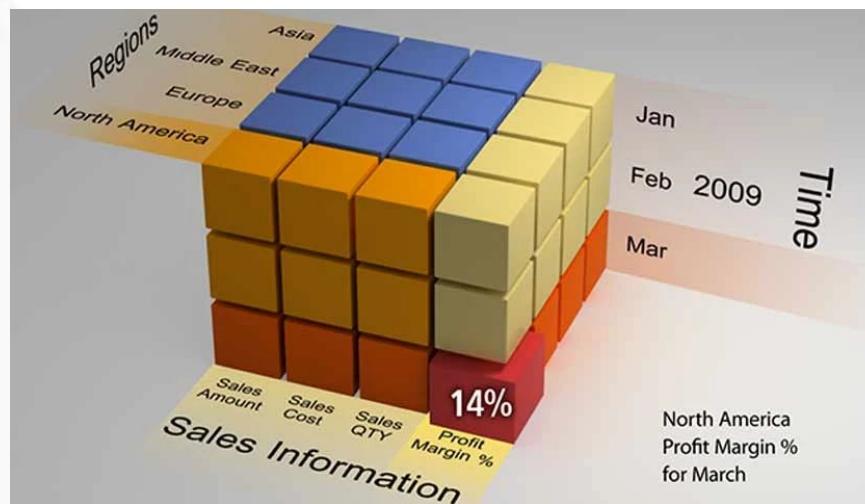
OLAP (Online Analytical Processing ou Processo Analítico em Tempo Real) é uma das ferramentas mais usadas para a exploração de um data warehouse. O OLAP possibilita alterar e analisar grandes quantidades de dados em várias perspectivas diferentes. As funções básicas do OLAP são:

- ✓ Visualização multidimensional dos dados.
- ✓ Exploração.
- ✓ Rotação.
- ✓ Vários modos de visualização.

OLAP

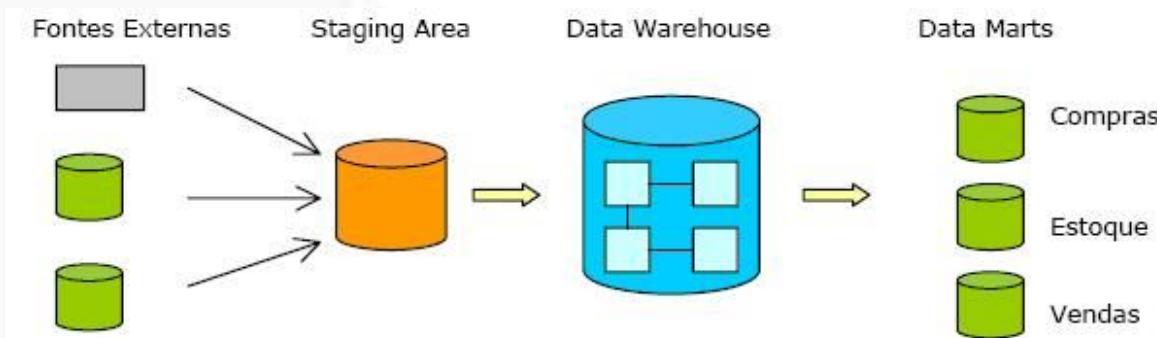
Online Analytical Processing

- ✓ OLAP e o Data Warehouse são destinados a trabalharem juntos, enquanto o DW armazena as informações de forma eficiente, o OLAP deve recuperá-las com a mesma eficiência, porém com muita rapidez.
- ✓ Um cubo OLAP é uma estrutura de dados montada de forma multidimensional, e que proporciona uma rápida análise de valores quantitativos ou medidas relacionadas com determinado assunto, sob diversas perspectivas diferentes.



Staging Area

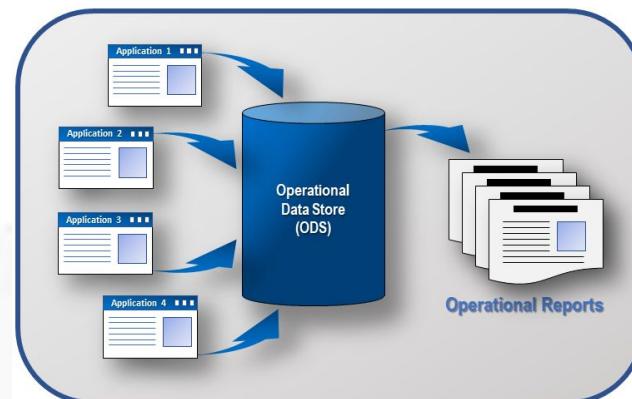
- ✓ A *Staging Area* é uma localização temporária onde os dados dos sistemas de origem são copiados, facilitando a integração dos dados antes de sua atualização DW. Tem como função agilizar o processo de consolidação, proporcionando um melhor desempenho na fase da atualização dos dados.
- ✓ A Staging Área é o único lugar para determinar os valores que vêm efetivamente dos sistemas legados. A Staging Área dever ser usada para limpeza dos dados que entram no processo de extração e transformação.



ODS – Operational Data Store



- ✓ ODS é um repositório de dados onde são colocados os dados que a empresa trabalha no seu dia a dia, para que sejam consultados por outros sistemas, ou por áreas de inteligência.
- ✓ Um ODS reúne dados de várias aplicações **e não é semelhante a um Data Warehouse, pois não tem o compromisso de armazenar histórico de dados e de servir para processos de auditoria sobre esses dados.**
- ✓ Entretanto o ODS deve armazenar dados que **tem “valor”** para seus consumidores e de manter-se atualizado.



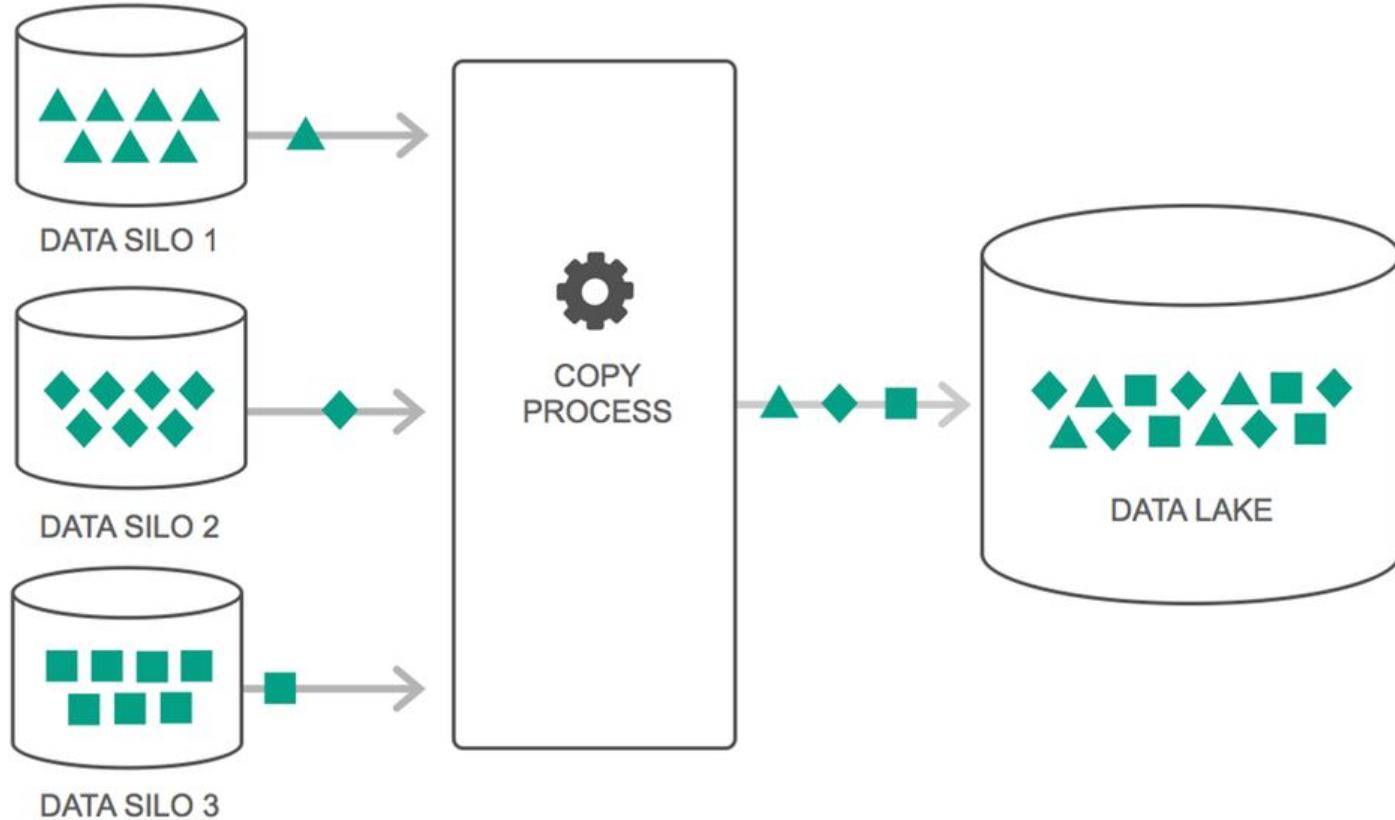
ODS



- ✓ Uma área de preparação normal destina-se apenas ao recebimento dos dados operacionais das origens transacionais, a fim de transformar os dados e carregá-los no armazém de dados.
- ✓ Um ODS também oferece essa funcionalidade, mas, além disso, pode ser consultada diretamente.
- ✓ Dessa forma, as ferramentas de análise que precisam de dados mais próximos do tempo real podem consultar os dados do ODS à medida que são recebidos dos respectivos sistemas de origem, antes de operações demoradas de transformação e carregamento.

Data Lake

IGTI



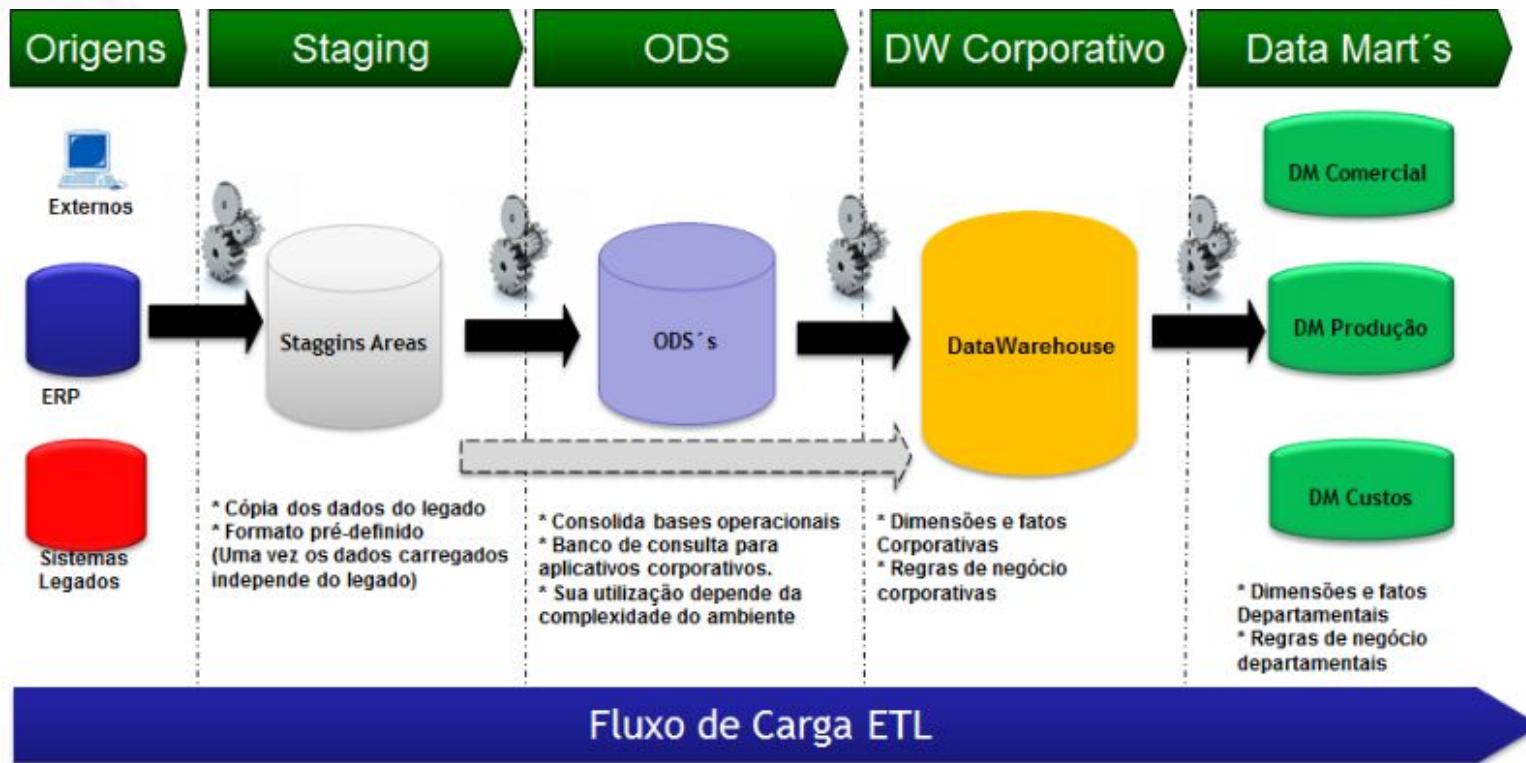
Data Lake

IGTI

	Data Warehouse	Data Lake
Dados	<ul style="list-style-type: none">· Estruturados· Processados	<ul style="list-style-type: none">· Estruturados / Semi-estruturados / Não estruturados· Não processados (em estado bruto)
Processamento	<ul style="list-style-type: none">· Esquema de dados gerado no momento da escrita	<ul style="list-style-type: none">· Esquema de dados gerado no momento da leitura
Armazenamento	<ul style="list-style-type: none">· Alto custo para alto volume de dados	<ul style="list-style-type: none">· Criado para ser de baixo custo, independente do volume de dados
Agilidade	<ul style="list-style-type: none">· Pouco ágil, configuração fixa	<ul style="list-style-type: none">· Bastante ágil, pode ser configurado e reconfigurado conforme necessário
Segurança	<ul style="list-style-type: none">· Estratégias de segurança bastante maduras	<ul style="list-style-type: none">· Ainda precisa aperfeiçoar o modelo de segurança e acesso aos dados
Usuários	<ul style="list-style-type: none">· Analistas de Negócios	<ul style="list-style-type: none">· Cientistas e Analistas de Dados

Processos do ETL

IGTI



Conclusão



Vimos os principais conceitos do processo ETL.

- ✓ Business Intelligence
- ✓ Data Warehouse (DW)
- ✓ Data Mart (DM)
- ✓ OLAP
- ✓ Staging Área
- ✓ Data Lake
- ✓ ODS

Próxima Aula



01. ••

Modelagem Dimensional

03. ••

02. ••

Modelos Star Schema e Snow
Flake

04. ••

Armazenamento de Dados

Capítulo 05 – Aula 05.02 – Modelagem Dimensional

PROF.: RICARDO BRITO ALVES

Nesta aula



- Modelagem Dimensional
- Tipos de Modelagem

Modelagem Dimensional

IGTI

- ✓ Técnica de modelagem de banco de dados para o auxílio às consultas do Data Warehouse, nas mais diferentes perspectivas, onde as informações se relacionam de maneira que podem ser representadas metaforicamente como um cubo.



Fonte: Nardi (2007)

Modelagem Dimensional



- ✓ Podemos fatiar este cubo e aprofundar em cada dimensão ou eixo para extrair mais detalhes sobre os processos internos que ocorrem na empresa que em um modelo relacional torna-se muito complicados de serem extraídos e muitas vezes até impossíveis de serem analisadas.
- ✓ O modelo dimensional permite visualizar dados abstratos de forma simples e relacionar informações de diferentes setores da empresa de forma muito eficaz.

Modelagem Dimensional

- ✓ A visão multidimensional permite o uso mais intuitivo para o processamento analítico pelas ferramentas OLAP (On-line Analytical Processing).
- ✓ Toda modelagem dimensional possui dois elementos imprescindíveis: **as tabelas Fatos e as tabelas Dimensões.** Ambas são obrigatórias e possuem característica complementares dentro de um Data Warehouse.

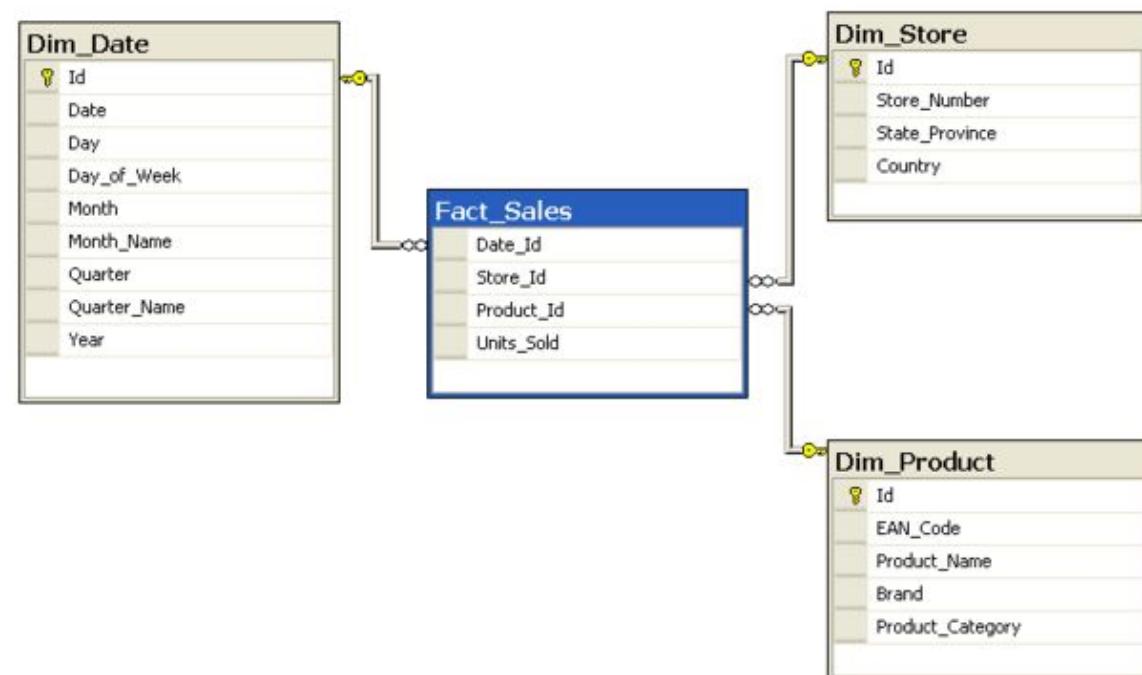


Modelo Star Schema

- ✓ O Star schema, idealizado por Ralph Kimball, é o modelo mais utilizado na modelagem dimensional para dar suporte à tomada de decisão e melhorar a performance de sistemas voltados para consulta.
- ✓ O Star Schema é composto no centro por uma tabela Fato, rodeada de dimensões, ficando parecido com a forma de uma estrela.
- ✓ Dessa forma as tabelas dimensionais devem conter todas as descrições que são necessárias para definir uma classe como Produto, Tempo ou Loja nela mesma, ou seja, as tabelas de dimensões não são normalizadas no modelo estrela, então campos como Categoria, Departamento, Marca contém suas descrições repetidas em cada registro, assim aumentando o tamanho das tabelas de dimensão por repetirem estas descrições de forma textual em todos os registros.

Modelo Star Schema

- ✓ Considerando um banco de dados de lojas, produtos e um data warehouse para executar relatórios de vendas agrupados por loja, data, ou categoria ou marca do produto. Se esse data mart estiver usando um esquema em estrela , teria a seguinte aparência:

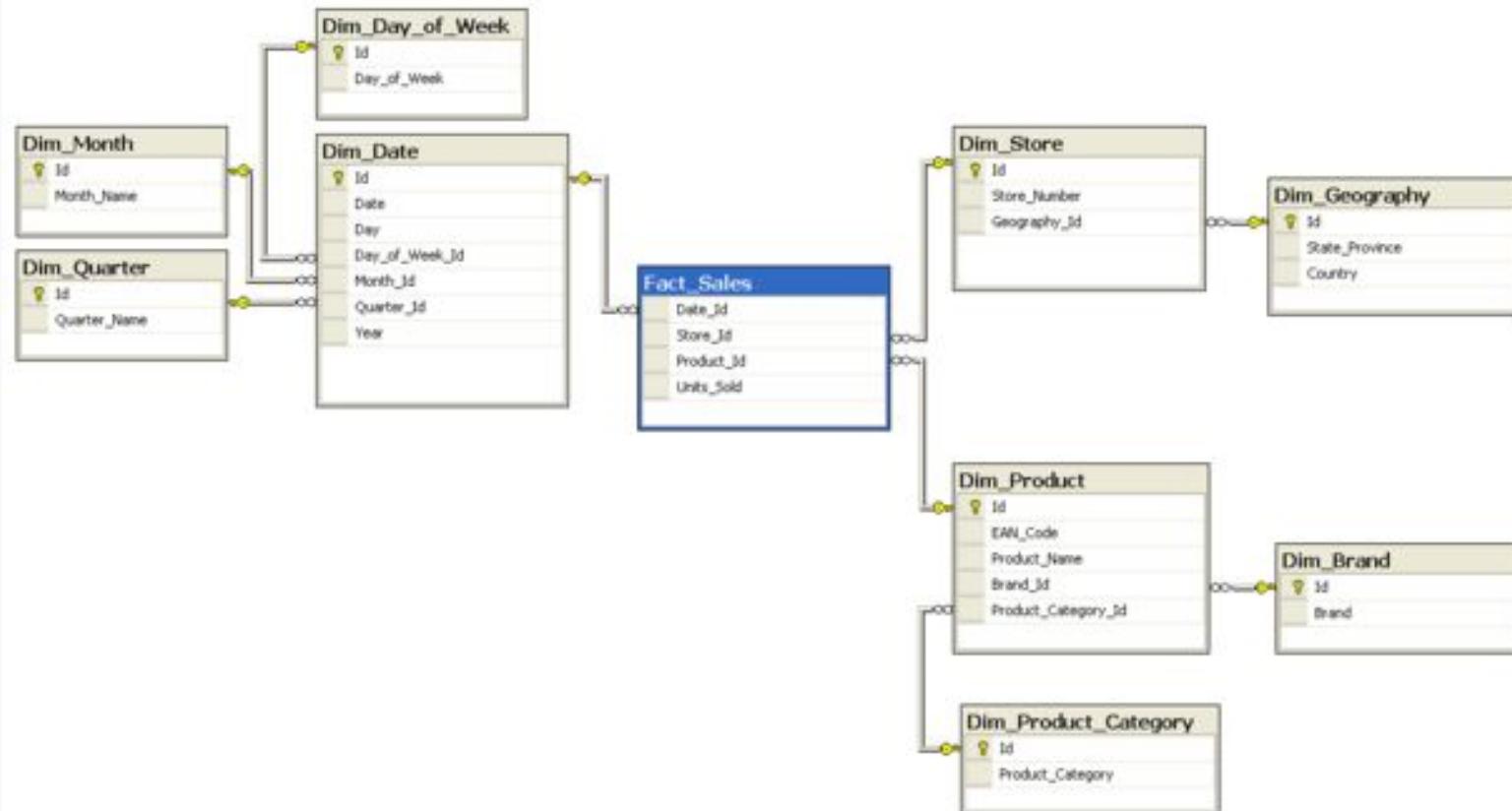


Modelo SnowFlake

- ✓ No modelo SnowFlake, defendido por Bill Inmon, as tabelas dimensionais relacionam-se com a tabela de fatos, mas algumas dimensões relacionam-se apenas entre elas. Isto ocorre para fins de **normalização** das tabelas dimensionais, visando diminuir o espaço ocupado por estas tabelas, então informações como Categoria, Departamento e Marca tornaram-se tabelas de dimensões auxiliares.
- ✓ Construindo a base de dados desta forma, passamos a utilizar mais tabelas para representar as mesmas dimensões, mas ocupando um espaço em disco menor do que o modelo estrela.
- ✓ Este modelo chama-se Snow Flake, pois cada dimensão se divide em várias outras tabelas, onde organizadas de certa forma lembra um flocos de neve.

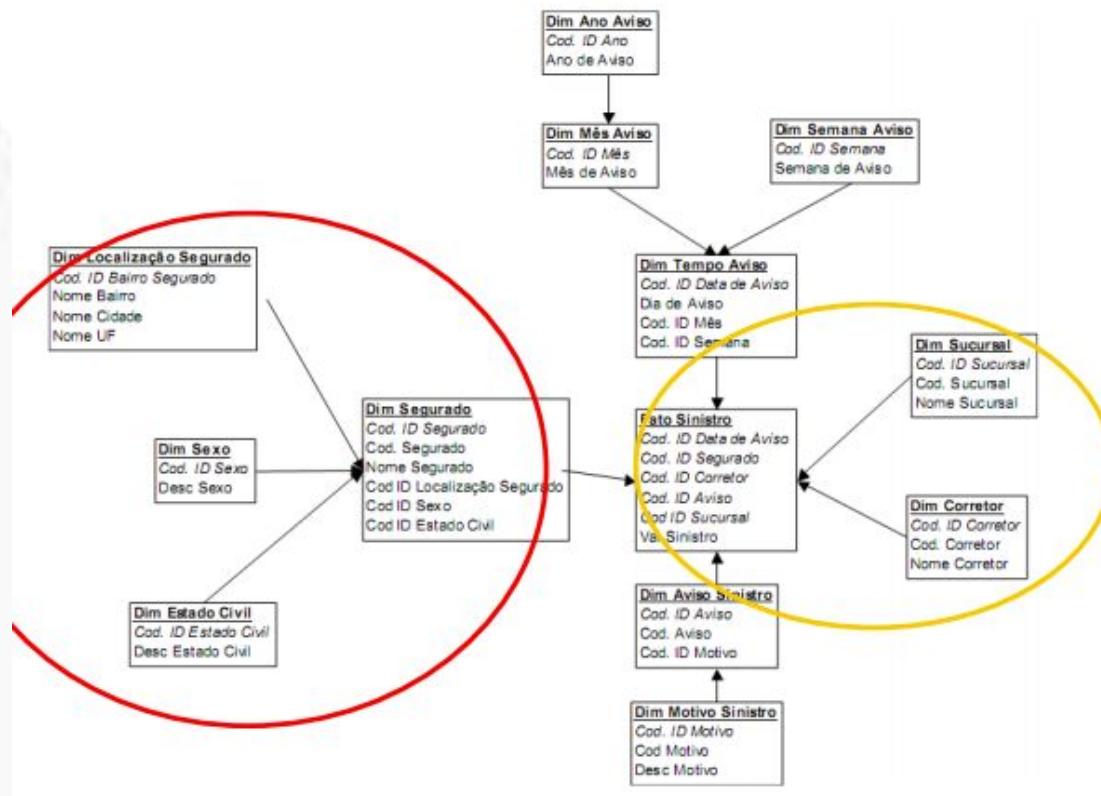
Modelo SnowFlake

- ✓ O mesmo cenário no esquema SnowFlake seria estruturado da seguinte maneira:



Modelo StarFlake

- ✓ É um híbrido entre os modelos Star Schema e Snowflake, aproveitando o melhor de cada um.



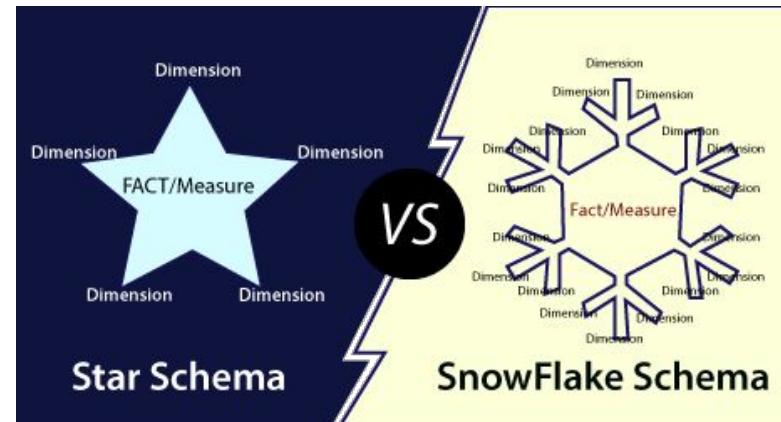
Considerações sobre os Modelos Dimensionais

✓ Modelos Star Schema (mais usado)

- Dimensões Desnormalizadas.
- Voltado para acessos com performance.
- Hierarquias achatadas.
- Mais simples e mais fácil navegação.
- Utiliza mais espaço repetindo as mesmas descrições ao logo de toda a tabela.

✓ Modelo Snowflake

- Normalizado
- Hierarquias mantidas.
- Muitas tabelas □ Muitas Junções – 1:N.
- Reduz o espaço de armazenamento dos dados dimensionais, mas acrescenta várias tabelas, deixando-o mais complexo.
- **Acesso mais lento que no StarSchema.**



Conclusão



- ✓ Modelagem Dimensional
- ✓ Modelos dimensionais StarSchema, SnowFlake e StarFlake

Próxima aula



01. ••

Dimensões

03. ••

02. ••

Extração, Transformação, Carga

04. ••

Armazenamento de Dados

Capítulo 05 – Aula 05.03 – Dimensões

PROF.: RICARDO BRITO ALVES

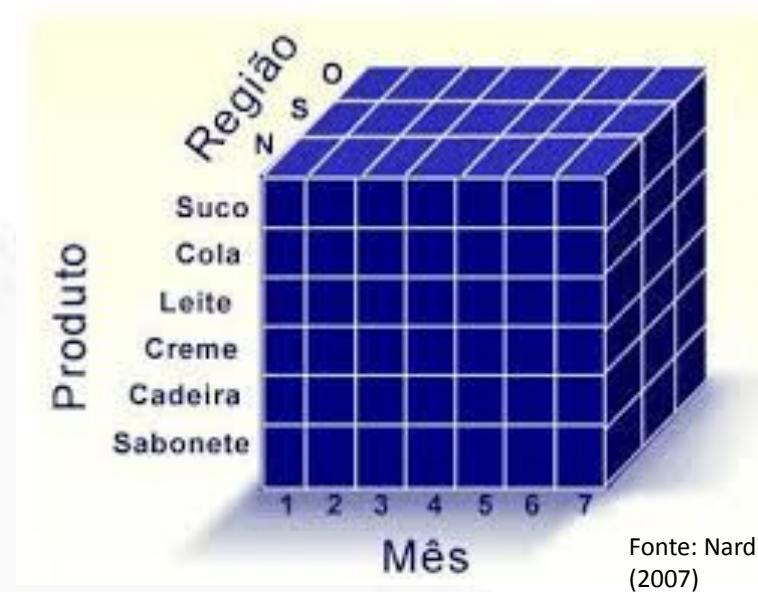
Nesta aula



- Tabela Dimensão
- Hierarquia
- Surrogate Key

Dimensão

- ✓ As dimensões identificam um indicador de análise sobre um empreendimento, negócio ou ação feita.
- ✓ Através das dimensões é possível identificar **quando** (mês, ano), **onde** (estado, região) e **com quem** (segurado, produto) ocorreu um indicador de análise (prêmio emitido).



Fonte: Nardi
(2007)

Tabela Dimensão

- ✓ A tabela dimensão tem como finalidade armazenar informações como tempo, geografia, produto, cliente.
- ✓ É comum uma tabela dimensão possuir várias colunas de informação com o objetivo de representar sua **hierarquia**.
- ✓ Sua interação com as **tabelas fato** é feita através da relação 1:N.
- ✓ Possuem uma chave primária para garantir a unicidade de seus registros e está presente na tabela fato, consequentemente como parte de sua chave primária.
- ✓ As dimensões armazenam 3 coisas:
 - **A Surrogate Key**
 - **A Natural Key**
 - **Os atributos**

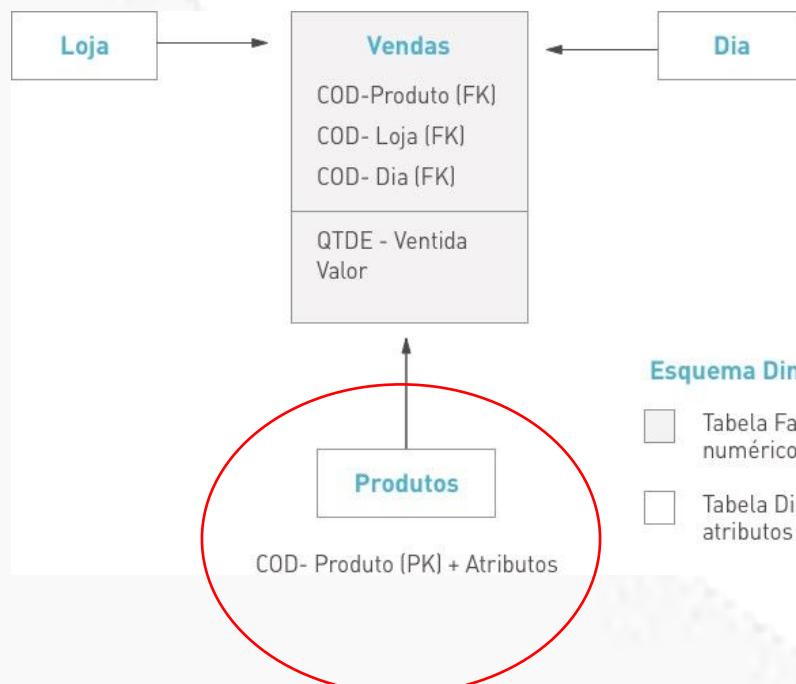
DIMENSÃO 1			
Surrogate Key (PK)	Natural Key	Atributo 1	Atributo 2
Chave artificial auto incremental	PK da origem / legado	Qualificadores da dimensão	

Fonte: rafaelPiton

Tabela Dimensão

Uma tabela de dimensão contém campos. Cada campo é chamado de “Atributo”.

O Produto poderia ser representado por uma tabela dimensão:



Esquema Dimensional

	Results	Messages
	ProdutOID	NomeProduto
1	121	Caderno
2	215	Lápis
3	347	Borracha
4	184	Calculadora

- Tabela Fato: chaves + fatos numéricos e aditivos
- Tabela Dimensão: chaves + atributos de filtros

Tabela Dimensão

- ✓ NomeProduto é um atributo, ProdutoID é um atributo-chave. Em um Data Warehouse, o atributo-chave em uma dimensão deve conter um valor exclusivo para cada membro da dimensão. Em um banco de dados relacional, este atributo-chave é denominado chave primária (PK).

	ProdutoID	NomeProduto
1	121	Caderno
2	215	Lápis
3	347	Borracha
4	184	Calculadora

Tabela Dimensão

- ✓ Além de tornar a tabela de Fato mais reduzida, mover informações sobre dimensão para uma tabela separada tem uma vantagem adicional: você pode adicionar novas informações sobre cada membro da dimensão. Exemplo: Categoria.
- ✓ Categoria agora é um atributo adicional da dimensão Produto. Se souber o *ProdutoID*, você poderá determinar não apenas *NomeProduto*, mas também a *Categoria* daquele Produto.

	Results	Messages	
	Produtoid	NomeProduto	Categoria
1	121	Caderno	Papelaria
2	215	Lápis	Papelaria
3	347	Borracha	Papelaria
4	184	Calculadora	Eletrônicos

Hierarquia de Dimensões



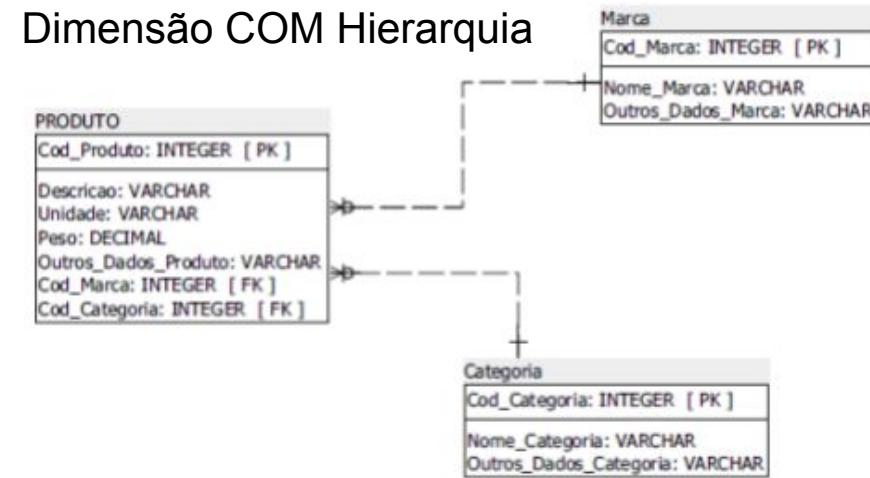
- ✓ É o conjunto de atributos que possuem uma ordem lógica do maior ao menor nível, é uma forma hierárquica de organizar os dados nas dimensões.
- ✓ Alguns exemplos de hierarquia:
 - ano, mês e dia.
 - categoria, subcategoria e produto.
 - capitão, sargento, cabo e soldado.
- ✓ As dimensões podem relacionar-se através de HIERARQUIAS. É comum existir apenas uma hierarquia por dimensão, mas podem existir múltiplas, se necessário.
- ✓ A hierarquia existe apenas nas dimensões, porque a métrica é só um valor, e quem vai dizer se esse valor está correspondendo a um determinado nível da hierarquia é o atributo.

Hierarquia de Dimensões

- ✓ O grão, ou detalhe, é o menor nível da hierarquia da dimensão. É a informação base, o menor detalhe da informação.
- ✓ É muito comum o cliente querer esse tipo de hierarquia para poder ir descendo ou subindo o nível da informação, é o que chamamos de **drill down e drill up**. Assim podemos ter uma visualização por ano e fazer um drill down, tendo a visualização por mês e depois por dia (que seria o grão).

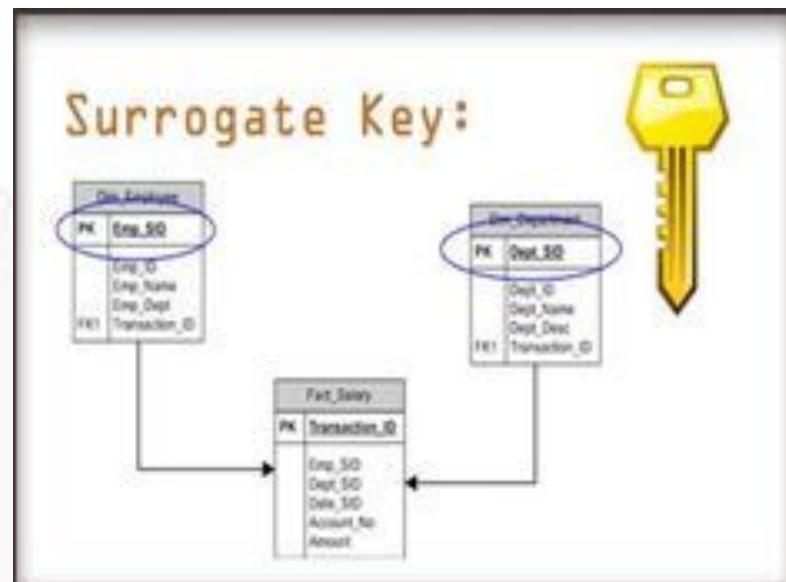
PRODUTO
Cod_Produto: INTEGER [PK]
Descricao: VARCHAR
Unidade: VARCHAR
Peso: DECIMAL
Outros_Dados_Produto: VARCHAR
Cod_Marca: INTEGER
Nome_Marca: VARCHAR
Outros_Dados_Marca: VARCHAR
Cod_Categoria: INTEGER
Nome_Categoria: VARCHAR
Outros_Dados_Categoria: VARCHAR

Dimensão SEM Hierarquia



Surrogate Key

- ✓ Em um banco de dados, as chaves são usadas para identificar as linhas de uma tabela e fazer as conexões entre elas. No Data Warehouse, temos a **Surrogate Key** nas dimensões, que é a **chave artificial** utilizada para conectar a tabela na Fato.
- ✓ A **Surrogate Key** nada mais é que a **Primary Key** da dimensão.



Surrogate Key

- ✓ A Surrogate Key é uma chave artificial e auto incremental.
- ✓ A palavra artificial vem do tipo, porque ela não existe em lugar nenhum, não está lá no transacional como a Natural Key (PK que vem do legado), ela é criada no Data Warehouse.
- ✓ E é auto incremental porque toda vez que é chamada, troca de número, então ela começa com 1 e vai indo para 2, 3, 4, e assim por diante.

Dimensão Produto							
SK	Chave Natural ou Chave de Negócios		Descrição do Produto	Peso	Unidade	Data Inicio	Data Fim
	14	1000	Chocolate X	180	Gramas	01/01/2017	
13	1000		Chocolate X	200	Gramas	01/01/2015	31/12/2016
12	1000		Chocolate X	210	Gramas	01/01/2010	31/10/2014
15	1002		Chocolate Y	170	Gramas	01/01/2017	
9	1002		Chocolate Y	200	Gramas	01/01/2010	31/12/2016

Surrogate Key

- ✓ Ela é gerada automaticamente na hora da carga, quando você carrega a dimensão no ETL.
- ✓ Na tabela Fato, essa **Surrogate Key** vai ser uma **Foreign Key**, a chave que serve para relacionar os dados entre duas tabelas, sempre apontando para uma Primary Key em outra tabela, que no caso da dimensão, vai ser a Surrogate Key.
- ✓ Assim, a tabela Fato receberá apenas o código da Surrogate Key da linha que ela está referenciando e não os atributos.

Surrogate Key

- ✓ Neste caso, a Surrogate Key da tabela dimensão **Estado** se relaciona com a FK da tabela Fato que armazena o código do estado. a Surrogate Key da tabela dimensão **Produto** se relaciona com a FK da tabela Fato que armazena o código do produto.

Dimensão Estado

Surrogate Key (PK)	Business Key	Descrição
1	MG	Minas Gerais
2	SP	São Paulo
3	RJ	Rio de Janeiro
4	ES	Espírito Santo

Dimensão Produto

Surrogate Key (PK)	Business Key	Descrição
1	PRD.001	Lápis
2	PRD.002	Papel
3	PRD.003	Caneta
4	PRD.004	Mochila
5	PRD.005	Caderno

Fato Vendas

SK Dimensão Estado (FK SK Dimensão Produto)	SK Dimensão Produto (FK Valor da Venda)	Valor da Venda
1	1	24,80
1	3	13,74
3	5	35,20
4	4	149,90
2	2	28,90

Surrogate Key

Resumindo, a Surrogate Key:

- ✓ Tem as características de uma Primary Key.
- ✓ É utilizada para referenciar a dimensão na Fato.
- ✓ É auto incremental.
- ✓ É uma chave artificial.
- ✓ É criada no Data Warehouse.
- ✓ Não pode se repetir.

Conclusão



- ✓ Tabela Dimensão.
- ✓ Hierarquia de Dimensões.
- ✓ Surrogate Key.

Próxima aula



01.

03.

02.

04.

Fato

Armazenamento de Dados

Capítulo 05 – Aula 05.04 – Fato

PROF.: RICARDO BRITO ALVES

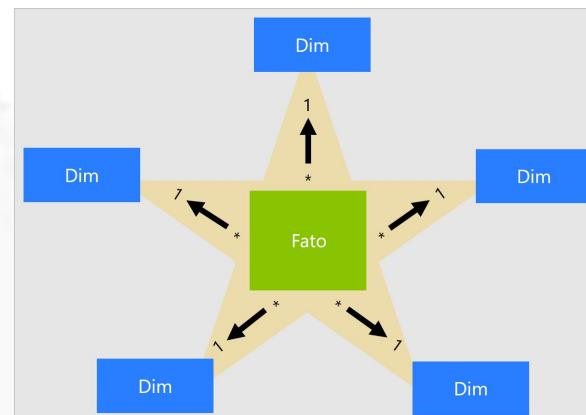
Nesta aula



- Tabela Fato
- Tipos de Tabelas Fato

Tabela Fato

- ✓ Principal tabela do Data Warehouse, ela vai se conectar nas dimensões.
- ✓ Podem existir uma ou mais tabelas fato.
- ✓ Armazenam principalmente:
 - **Métricas** - que são os fatos propriamente ditos (tudo que a empresa for mensurar é uma métrica).
 - **Foreign key** – chave estrangeira que serve para relacionar os dados das Dimensões com a Fato.



Tipos de Tabelas Fato



- ✓ As tabelas Fatos são classificadas pelas suas granularidades.
- ✓ Independente de sua granularidade, cada métrica em uma tabela Fato deve estar exatamente no mesmo nível de detalhe.
- ✓ Existem 6 tipos de fatos:
 - Fato transacional.
 - Fato agregada.
 - Fato consolidada.
 - Fato snapshot periódico.
 - Fato de snapshot acumulado.
 - Fato sem Fato.

Fato Transacional

- ✓ Fatos transacionais são as mais comuns.
 - ✓ Geralmente utilizam **métricas aditivas**, aquelas que somam por todas as Dimensões.
 - ✓ Há 2 formas de armazenar os dados em uma tabela Fato transacional.
 - Em uma forma de transação por linha.
 - Em uma forma de linha por transação, é a mais utilizada.

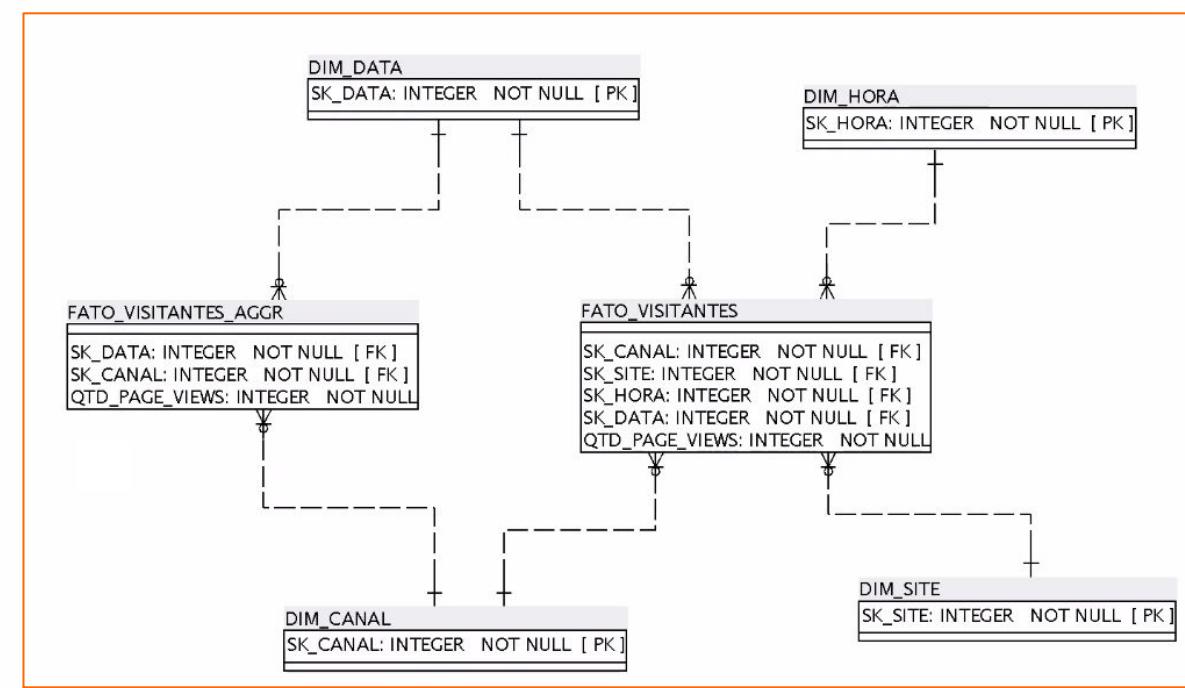
Fato Transacional (Vendas)										
Cruzamento/Dimensionalidade é sempre feito no menor nível da transação (Grão. Nível 0.)										Métrica ativa
	DIM_DATA	DIM_PRODUTO	DIM_CLIENTE	DIM_GEOGRAFIA	DIM_VENDEDOR	#DD_TRANS	VL_PRECO_UNI	QTD_VENDA	VL_DESCONTO	VL_VENDA
Uma TRANSAÇÃO por Linha	10/12/2016	Coca-Cola	Piton	Porto Alegre	Bia	#4474343	3,8	1	0,00	3,80
	10/12/2016	Coca-Cola	Piton	Porto Alegre	Bia	#4474343	3,8	1	0,00	3,80
	10/12/2016	Coca-Cola	Piton	Porto Alegre	Bia	#4474343	3,8	1	0,00	3,80
	10/12/2016	Coca-Cola	Piton	Porto Alegre	Bia	#9988574	3,8	1	1,00	2,80

Fato Agregada

- ✓ Serve para juntar uma grande quantidade de dados quando não precisar analisar no nível do grão, com a função de acelerar o desempenho das consultas.
- ✓ Cria uma Fato agregada com os dados que precisa e poderia permanecer com uma Fato normal e outra agregada.

Fato Agregada

Exemplo: Uma tabela Fato visitantes e Dimensões de data e hora de um determinado site. A Fato vai monitorar sempre que alguém acessa o site, mas não os visitantes por minuto, porém posso precisar ver de forma consolidada.



Fato Consolidada

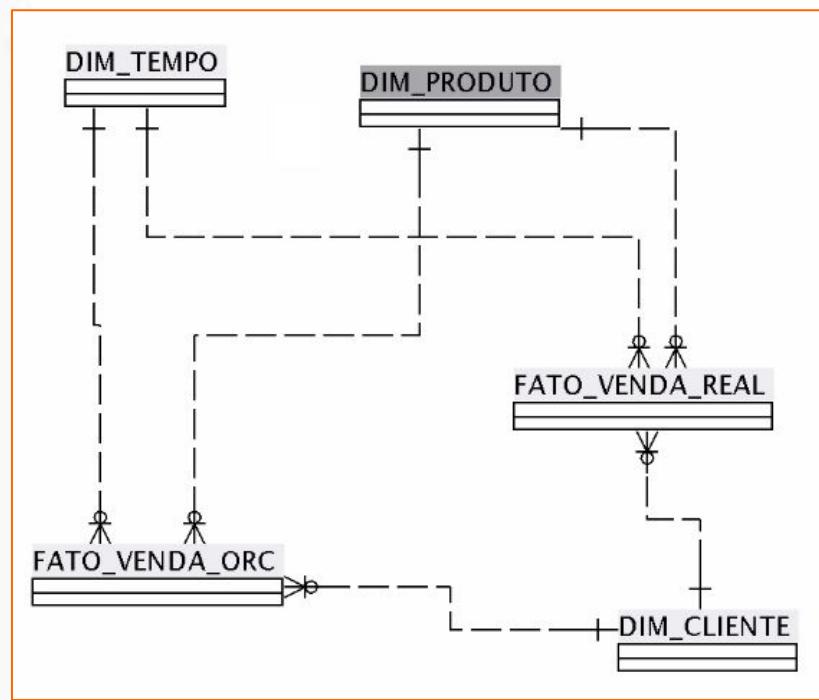


- ✓ É bem parecida com a agregada, mas serve para combinar 2 tipos de processos (área de negócio, área de assunto, processo de negócio).
- ✓ A Fato consolidada serve para consolidar duas tabelas Fato.
- ✓ No processamento do ETL, na hora de carregar a tabela Fato, carrega uma, carrega a outra, e mistura as duas. O grão precisa ser o mesmo entre as duas.

Fato Consolidada

Exemplo: tem uma Fato Venda, e depois precisa juntar ela com uma Fato Venda Orçada.

Depois disso, cria uma Fato Consolidada e coloca nela o valor real e o orçado, que antes estavam em fatos diferentes.



Fato Snapshot Periódico

IGTI

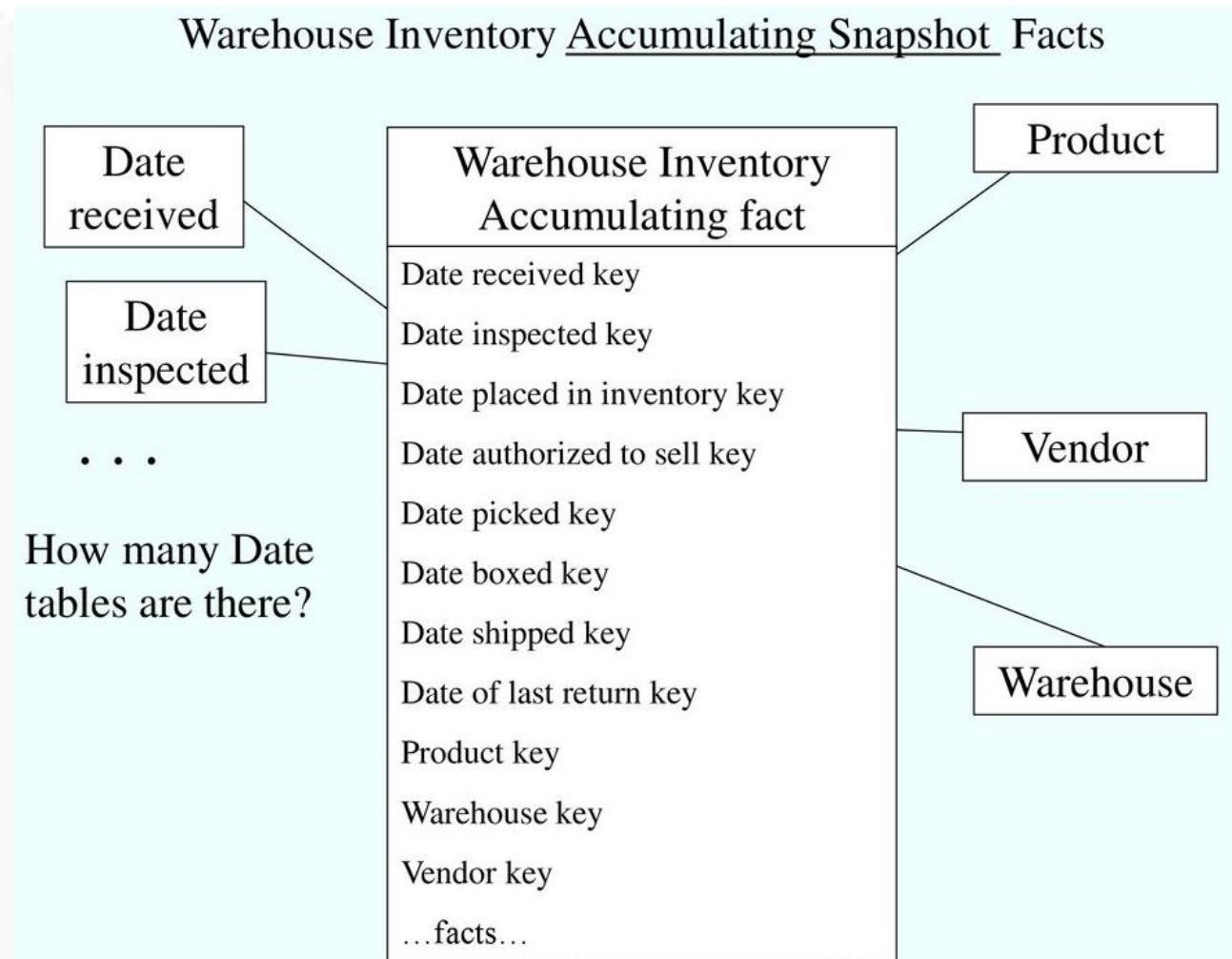
Fato_estoque_hora_transacional					
data	produto	qtd_entrada	qtd_saida	saldo	
23/10/2017 06:00:00	Café Ristreto	2.000	500	1.500	
23/10/2017 12:00:00	Café Ristreto	400	1.500	400	
23/10/2017 16:00:00	Café Ristreto	700	300	800	
...	

Fato_estoque_diaria_snapshot_periodica					
data	produto	qtd_entrada (opcional)	qtd_saida (opcional)	saldo	
23/10/2017	Café Ristreto	3.100	2.300	800	

O que é o snapshot periódico, então? Ele é baseado no tempo, seja data, dia, semana ou hora. Nesse exemplo a foto é ao final do dia.

Fato de Snapshot Acumulado

IGTI



Fato de Snapshots

IGTI



Bus Matrix



Atividades de Negócio	Dimensões Padronizadas													
	Data	Produto	Loja	Promoção	Vendedor	Fornecedor								
Vendas	X	X	X	X										
Inventário	X	X	X											
Entrega	X	X	X											
Compras	X	X								X		X		

Labor & Payroll	X							X	X	X				
Computer Charges	X	X	X		X			X	X	X	X	X	X	X
Purchase Orders	X							X	X	X	X	X	X	X
Supplier Deliveries	X							X	X	X	X	X	X	X

Fato sem Fato

Seria uma tabela Fato sem métricas.

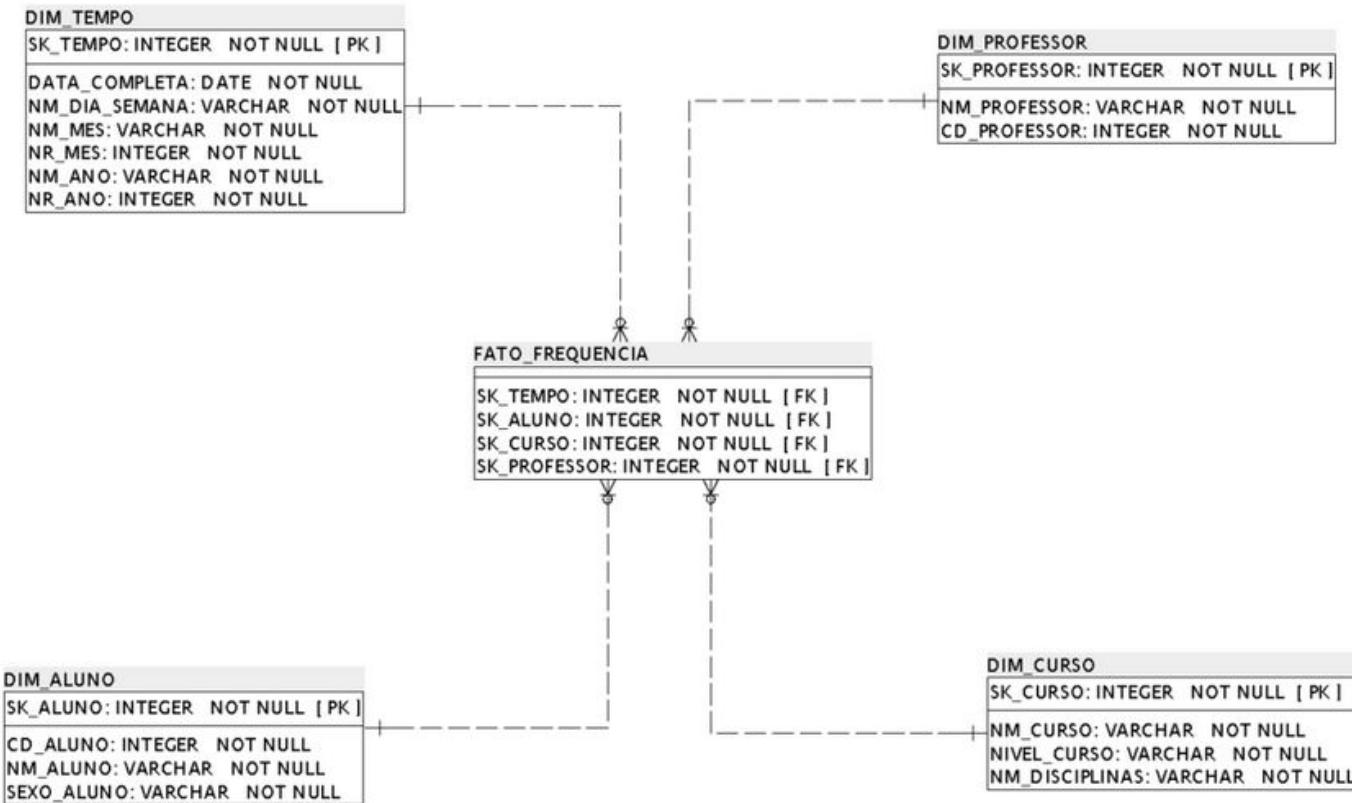
Ela também é chamada de Fato de Associação ou de Intersecção, mas o termo técnico é **Fato sem Fato ou Factless Fact Table**.

Serve para fazer uma intersecção de Dimensões. Às vezes a gente quer comparar ou cruzar algo somente entre duas ou mais Dimensões e não tem uma métrica para fazer essas comparações.

Essa tabela Fato é exceção, só é usada quando se precisa fazer uma intersecção entre as Dimensões.

Fato sem Fato

Temos o aluno, a universidade, o curso, um professor e a dimensão de tempo. Não temos uma métrica nesse caso.



Conclusão

- ❑ Tabela Fato
- ❑ Tipos de Tabelas Fato
 - Fato transacional.
 - Fato agregada.
 - Fato consolidada.
 - Fato snapshot periódico.
 - Fato de snapshot acumulado.
 - Fato sem Fato.

Próxima aula



01. ••

Definição de ETL

02. ••

03. ••

04. ••

Armazenamento de Dados

Capítulo 05 – Aula 05.05 – Definição de ETL

PROF.: RICARDO BRITO ALVES

Nesta aula

- Definição de ETL
- Como surgiu o ETL
- Extração
- Transformação
- Carga

O que é ETL?

Na fase de integração dos dados executamos o ETL.

Essa sigla significa **Extract, Transform and Load**, ou seja, Extração, Transformação e Carga.

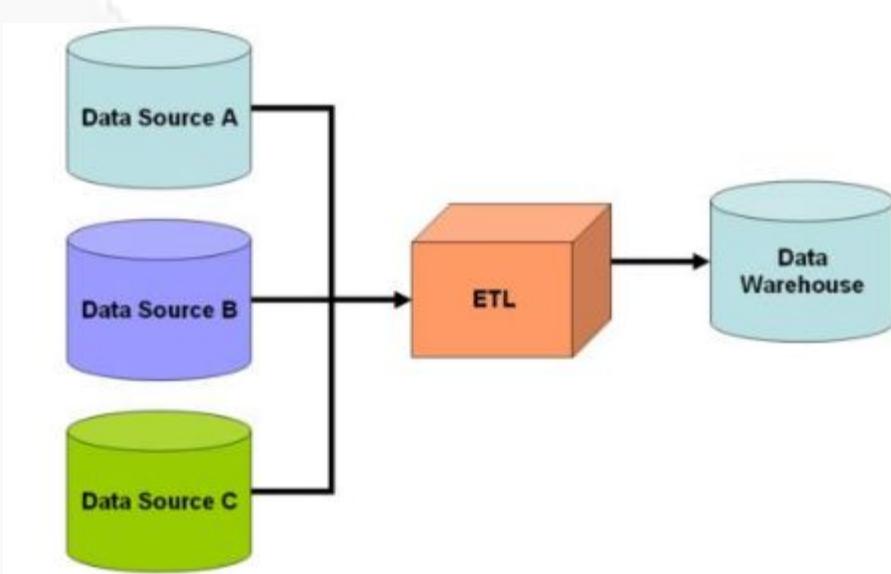
Comumente utilizado para construir um Data Warehouse. Nesse processo, os dados são retirados (extraídos) de um ou mais sistemas-fonte ou origens, convertidos (transformados) em um formato que possa ser analisado, e armazenados (carregados) em um armazém ou outro sistema.



O que é ETL?

IGTI

“Um sistema ETL projetado adequadamente extrai dados dos sistemas de origem, aplica padrões de qualidade e consistência de dados, alinha os dados para que fontes separadas possam ser usadas em conjunto e, finalmente, entrega dados em um formato pronto para apresentação, para que os desenvolvedores possam criar aplicativos e os usuários finais podem tomar decisões” ...*Ralph Kimball*



Como surgiu o ETL



- ✓ ETL ganhou popularidade nos anos 1970, com a necessidade de integrar os dados que se espalhavam por diferentes databases dentro de uma organização. O ETL tornou-se o método padrão para coletar dados de fontes diferentes e transformá-los antes de carregá-los no sistema destino.
- ✓ No início de 1990, os Data Warehouses entraram em cena, fornecendo um acesso integrado a dados de múltiplos sistemas.
- ✓ Com o tempo, a quantidade de fontes e sistemas de dados aumentou muito. Assim, o ETL virou um método padronizado para a coleta de dados, de diferentes fontes, e que foi aperfeiçoado ao longo das décadas seguintes. Em especial, após o boom da transformação digital.

ETL- Extração

- ✓ A extração é a primeira etapa de um processo de ETL.
- ✓ Os projetos de Data Warehouse consolidam dados de diferentes fontes.
- ✓ Um processo ETL precisa ser capaz de se comunicar com bases de dados e ler diversos formatos de arquivos utilizados por toda a organização.
- ✓ O primeiro passo é definir as fontes de extração e os dados podem vir das mais diversas fontes, por exemplo:
 - Sistemas de gestão (SIG, ERP, CRM etc).
 - Diversos SGBD's (Oracle, SQLSERVER, DB2 etc).
 - Arquivos como planilhas do Excel e documentos de texto.

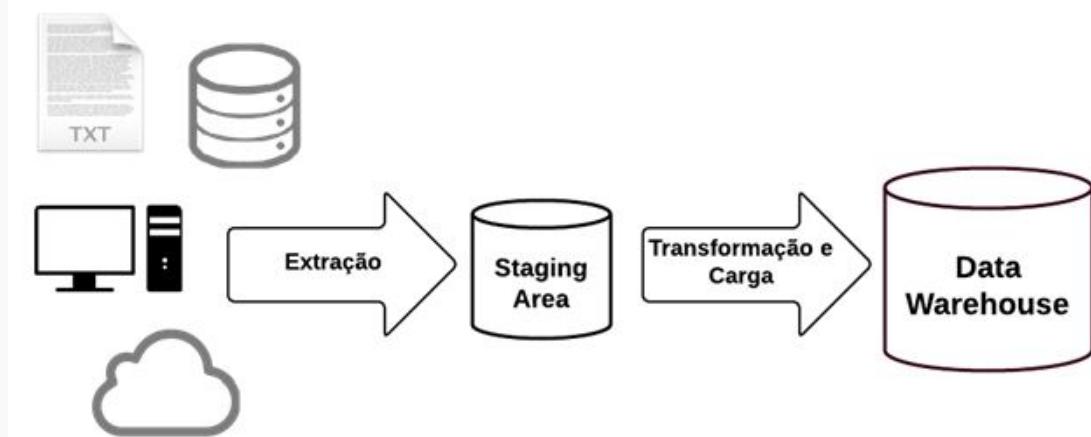
ETL- Ferramentas



- ✓ É importante definir a ferramenta que irá fazer a extração, por exemplo:
 - IBM Information Server (Data Stage);
 - Oracle Data Integrator (ODI);
 - Informática Power Center;
 - Microsoft Integration Services (SSIS).
- ✓ Existe também um conjunto de Ferramentas de ETL Open Source como o PDI – Pentaho Data Integrator e Talend ETL.

ETL - Extração

A etapa de extração é a fase onde os dados são extraídos dos OLTPs (Online Transaction Processing) e conduzidos para a staging area (área temporária), onde são convertidos para um único formato. A conversão se faz necessária devido a heterogeneidade existente nas informações oriundas desses sistemas, sendo essencial a conformação prévia para o tratamento adequado.



ETL - Transformação



- ✓ Após a extração, teremos subsídios para iniciar a etapa de transformação e limpeza dos dados. Nessa fase são corrigidos, padronizados e tratados os desvios e inconsistências, transformando os dados de acordo com as regras do negócio.
- ✓ É nesta etapa que realizamos os devidos ajustes, podendo assim, melhorar a qualidade dos dados e consolidar dados de duas ou mais fontes.
- ✓ O estágio de transformação aplica uma série de regras ou funções aos dados extraídos para ajustar os dados a serem carregados.

Operações de Transformação



O processo de transformação de dados é composto por vários subprocessos como por exemplo:

- ✓ **Limpeza** - inconsistências e valores ausentes nos dados são resolvidos.
- ✓ **Padronização** - regra de formatação é aplicada ao conjunto de dados.
- ✓ **Eliminar duplicados** - dados redundantes são excluídos ou descartados.
- ✓ **Verificação** - dados inutilizáveis são removidos e anomalias são sinalizadas.
- ✓ **Classificação** - os dados são organizados de acordo com o tipo.
- ✓ **Outras tarefas** - quaisquer regras adicionais / opcionais podem ser aplicadas para melhorar a qualidade dos dados.

ETL - Carga

- ✓ A etapa de carga ocorre em sequência com a de transformação.
Assim que são efetuados os tratamentos necessários nos dados, a carga no DW é iniciada. Essa fase se resume na persistência dos dados na base consolidada.
- ✓ Dentro de um mesmo DW temos diferentes períodos de execução para cada tipo de processo de carga. Alguns são mensais, outros diários.
- ✓ Neste momento também é definida a LATÊNCIA das informações.
Isso pode variar para cada tabela a ser carregada. Latência é sinônimo de atraso, é uma expressão de quanto tempo leva para um pacote de dados ir de um ponto designado para o outro.
- ✓ As cargas podem ser full ou incrementais.

Conclusão



Definição de ETL

Próxima Aula



01. ••

A importância do ETL

02. ••

Benefícios do ETL

03. ••

04. ••

Armazenamento de Dados

Capítulo 05 – Aula 05.06 – Importância do ETL

PROF.: RICARDO BRITO ALVES

Nesta aula

- Importância do ETL
- Importância do ETL nas empresas
- Dependência do BI do ETL
- Benefícios do ETL

Importância do ETL

IGTI

Há anos, inúmeras empresas têm confiado no processo de ETL para obter uma visão consolidada dos dados que geram as melhores decisões de negócios. Hoje, esse método de integrar dados de múltiplos sistemas e fontes ainda é um componente central do kit de ferramentas de data integration de uma organização.



Importância do ETL



O processo ETL é uma das fases mais críticas na construção de um sistema DW, visto que:

- ✓ Grandes volumes de dados são processados.
- ✓ Serão implementadas as regras e fórmulas dos indicadores que irão compor as tabelas de Fato.
- ✓ É essencial para a criação das estruturas de Dimensões e Fatos no ambiente do DW. É ele que faz a “ponte” de ligação entre o operacional e o DW.

Importância do ETL



- ✓ Devemos escolher bem as ferramentas que darão suporte ao processo, pois são essenciais para a correta execução das atividades do ETL.
- ✓ O ETL é fundamental para qualquer iniciativa de DW. Porém, deve ser planejado com cuidado para não comprometer os sistemas transacionais (OLTP) das empresas. Um bom ETL deve ter escalabilidade e ser passivo de ser mantido.
- ✓ Devemos analisar a janela de operação do ETL. Não é em qualquer momento que ele poderá ser executado.
- ✓ Devemos analisar a periodicidade de execução.

Esses detalhes são críticos para o sucesso do processo.

Importância do ETL



- ✓ Estudos relatam que o ETL e as ferramentas de limpeza de dados consomem um terço do orçamento num projeto de DW, podendo, no que tange ao tempo de desenvolvimento de um projeto de DW, chegar a consumir 80% desse valor.
- ✓ Outros estudos mencionam, ainda, que o processo de ETL consome 55% do tempo total de execução do projeto de DW.

Importância Nas Empresas



- ✓ Quando utilizado em um Data Warehouse corporativo, *o ETL fornece o contexto histórico completo para a empresa.*
- ✓ Ao fornecer uma visão consolidada, *o ETL facilita para os usuários corporativos a análise e a criação de relatórios sobre dados relevantes às suas iniciativas.*
- ✓ *O ETL pode melhorar a produtividade de profissionais analíticos,* porque ele codifica e reutiliza processos que movem os dados sem que esses profissionais possuam a capacidade técnica de escrever códigos ou scripts.

Importância Nas Empresas



- ✓ O ETL evoluiu ao longo do tempo para suportar os requisitos emergentes de integração para coisas como streaming data.
- ✓ As ferramentas de ETL podem ser utilizadas para fazer todo tipo de trabalho de importação, exportação, transformação de dados para outros ambientes de banco de dados ou para outras necessidades a serem endereçadas. Exemplo: Importação de base de outra empresa.

Importância do ETL para BI

IGTI



Benefícios ETL



✓ Histórico dos dados

Ao reunir as informações do negócio em um repositório, o ETL fornece um histórico da empresa e auxilia a produção de relatórios e análises sobre dados relevantes para iniciativas corporativas.

✓ Fácil de usar

Ferramentas de ETL eliminam a necessidade de programar o código para extrair e processar dados. Isso já é feito pela ferramenta, basta especificar as fontes dos dados e as regras para a sua transformação.

Benefícios ETL

✓ Funções avançadas para categorizar e limpar dados

Ferramentas ETL possuem muitas funções para auxiliar na limpeza e categorização de dados. Além disso, o uso dessas ferramentas apresenta vantagens na transferência e transformação de um grande volume de dados com regras complexas, simplificando seu cálculo, mudança e integração.

✓ Metainformação

Os metadados, ou seja, informações sobre como identificar, localizar e compreender os dados, são gerados automaticamente pelas aplicações ETL. Isso reduz falhas e auxilia a administração desses dados.

Conclusão



- ✓ Importância do ETL

Próxima Aula



01. ••

03. ••

ELT - Extract, Load, Transform

02. ••

04. ••

Armazenamento de Dados

Capítulo 05 – Aula 05.07 – Definição ELT

PROF.: RICARDO BRITO ALVES

Nesta aula



- Definição ELT – Extract, Load, Transform
- Diferenças entre ETL e ELT

Definição ELT



Extract – Extrair

Load – Carregar

Transform – Transformar

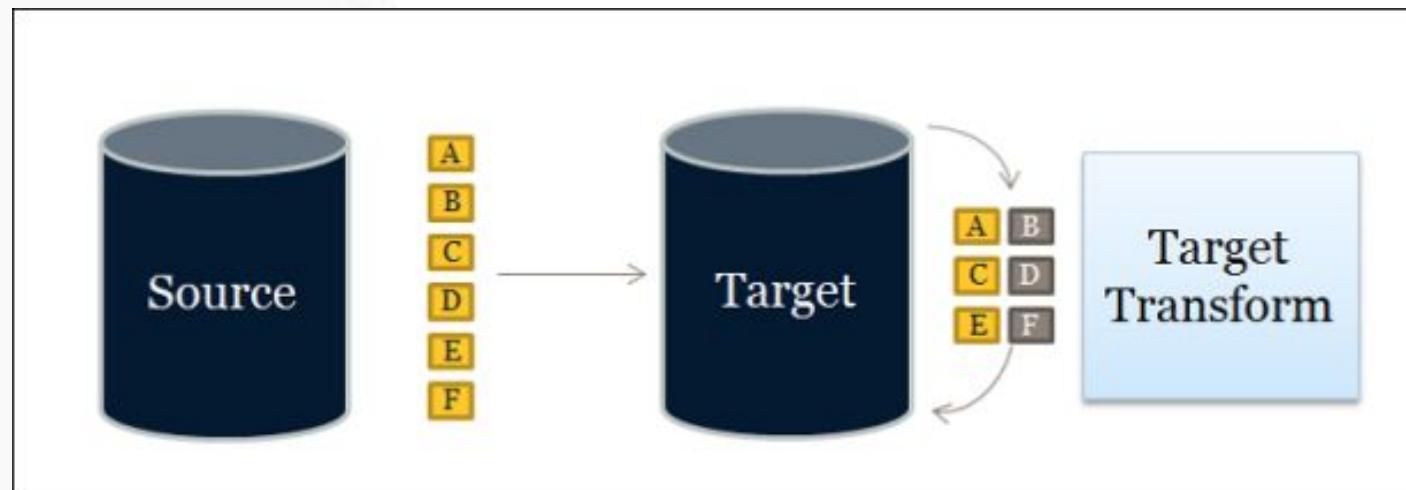
O **ELT** é um processo de dados usado para replicar dados de uma fonte para um banco de dados de destino, sendo uma evolução ETL, pois torna o processo de replicação de dados muito menos complexo, uma vez que o passo de transformação é realizado após os dados estarem no destino.



Passos do ELT

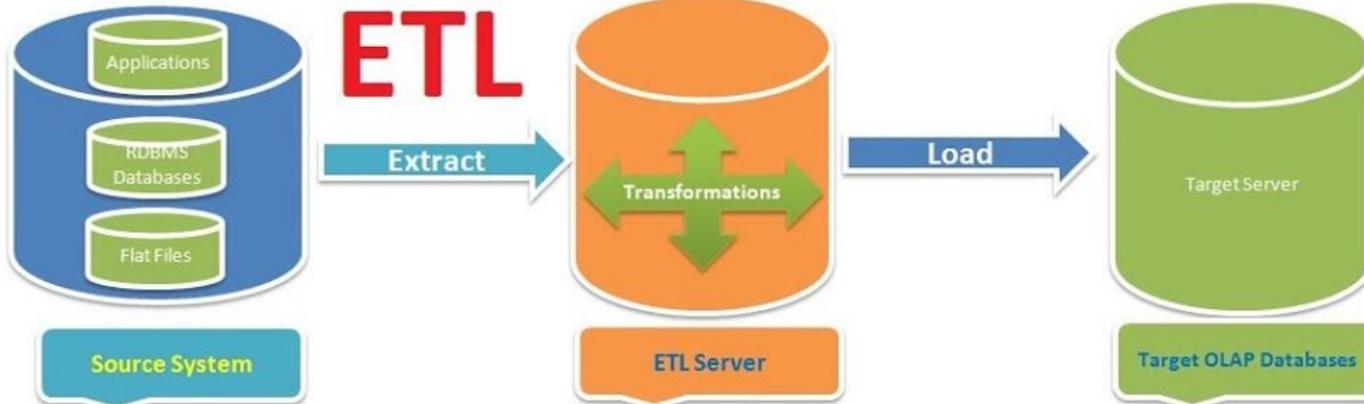
Os passos do ELT:

- Extrair: Copiar dados de uma fonte de origem.
- Carregar: Replicar esses dados no banco de dados de destino.
- Transformar: Modelar os dados para estruturá-los, facilitar consultas, cruzar com outras fontes e analisá-los.



Passos do ELT

IGTI



Principais diferenças entre ETL e ELT

- **ETL** – Extrair, Transformar e Carregar
- **ELT** – Extrair, Carregar e Transformar
- O **ETL** carrega os dados primeiro no ambiente intermediário (**Stage**) caso exista e posteriormente no destino, enquanto o **ELT** carrega os dados diretamente no destino.
- O **ETL** é usado para dados relacionais, estruturados e em ambientes on-premise ou nuvem, enquanto **ELT** é usado em ambientes de cloud (nuvem) com estruturas escalonáveis e fontes de dados não estruturadas.

Principais diferenças entre ETL e ELT

- O ETL não fornece suporte ao data lake, enquanto o **ELT** fornece suporte ao data lake.
- O ETL é mais fácil de implementar, enquanto o **ELT** se torna mais complexo para implementar e manter.

O ELT requer mais recursos do destino de dados, já que é lá que são realizadas as transformações de dados. No entanto, hoje o poder dos destinos de dados disponíveis em nuvem torna o ELT uma opção mais simples e ágil ao ETL e a escolha perfeita para as áreas de dados mais dinâmicas.

Benefícios ELT

- **Tempo:** uma vez que os dados são extraídos, podem ser carregados no Data Lake imediatamente. Os passos de conversão podem ou não ser implementados posteriormente, mas os dados no Data Lake estão prontos para uso economizando tempo.
- **Custo:** Como o destino é geralmente uma solução NoSQL e este tipo de solução tem custos de licenciamento baixo, mostra-se uma solução de bom custo-benefício.
- **Flexibilidade:** Este tipo de solução é extremamente útil para análises AD-HOC e sua (aparente) simplicidade torna esta técnica próxima de desenvolvedores ou até mesmo consumidores de dados de baixo nível técnico.

Conclusão



- ✓ Definição do ELT

Próxima Aula



01.

03.

02.

04.

Práticas com Pentaho PDI

Armazenamento de Dados

Capítulo 06 – Práticas com Pentaho PDI

PROF.: RICARDO BRITO ALVES

Armazenamento de Dados

Capítulo 06 – Aula 06.01 – Práticas com Pentaho –
Transformações - Inputs

PROF.: RICARDO BRITO ALVES

Armazenamento de Dados

Capítulo 06 – Aula 06.02 – Práticas com Pentaho –
Transformações - Outputs

PROF.: RICARDO BRITO ALVES

Armazenamento de Dados

Capítulo 06 – Aula 06.03 – Práticas com Pentaho –
Transformações de Dados

PROF.: RICARDO BRITO ALVES

Armazenamento de Dados

Capítulo 06 – Aula 06.04 – Steps de um DW

PROF.: RICARDO BRITO ALVES