

BANCO DE DADOS

Autor: Leonardo Costa Passos

Resumo com o que há de mais importante da disciplina de BANCO DE DADOS para provas de Concursos Públicos com foco na banca CESGRANRIO.

Se o conteúdo for útil na sua jornada de estudos, você pode me agradecer fazendo um PIX de um valor que considere justo para a seguinte chave:

leonardx@gmail.com

Table of Contents

1.	MODELAGEM CONCEITUAL DE DADOS.....	5
1.1.	Níveis de Abstração.....	5
1.2.	Independência de Dados	5
1.3.	Arquitetura de Três Níveis	6
2.	MODELO ENTIDADE RELACIONAMENTO (MODELO CONCEITUAL)	7
2.1.	Atributos	7
2.2.	Relacionamentos.....	10
2.1.1	GRAU DE RELACIONAMENTO:	10
2.1.2	CARDINALIDADE DE RELACIONAMENTO:	11
2.3.	Entidades.....	14
2.3.1	Generalização/Especialização	14
2.3.2	Entidade Fraca.....	16
2.3.3	Entidade Associativa	17
2.4.	Diagrama Entidade – Relacionamento (DER).....	18
2.5.	Notação James Martin (ou Pé de Galinha – Crow’s Foot)	19
3.	SGBD (Conceitos)	20
3.1.	BD Operacional x Data Warehouse:.....	21
3.2.	Sistemas de Gerenciamento de Banco de Dados (SGBDs)	21
3.3.	Esquema X Instância:	22
3.4.	Metadados de Arquivos	23
3.5.	Dicionário/Catálogo de Dados	24
4.	Linguagem SQL.....	25
4.1.	Introdução.....	25
4.2.	Subconjuntos da Linguagem SQL.....	26
4.3.	Sintaxe Básica.....	27
4.4.	A instrução Select:	28
4.5.	As Instruções JOIN.....	29
4.6.	A Instrução CREATE.....	35
4.6.1	Create Table	35
4.6.2	Create View	38
4.6.3	Create TRIGGER.....	38
4.7.	A Instrução INSERT.....	39
4.8.	A Instrução UPDATE	41
4.9.	A Instrução DELETE	42
4.10.	A Instrução GRANT.....	42

4.11.	A Instrução REVOKE	42
4.12.	Restrições SQL.....	43
5.	DADO, INFORMAÇÃO, CONHECIMENTO E INTELIGÊNCIA	43
5.1.	Dados:	43
5.2.	Informação:.....	43
5.3.	Conhecimento.....	44
5.4.	Inteligência.....	45
6.	BUSINESS INTELLIGENCE	46
7.	DATA WAREHOUSE	47
7.1.	Características de Datawarehouse	49
7.2.	Formas de Armazenamento OLAP	51
8.	MODELO MULTIDIMENSIONAL	51
10.1.	TIPOS DE FATOS NUMÉRICOS	55
10.2.	ESQUEMAS MULTIDIMENSIONAIS	55
10.2.1	Modelo Estrela	55
10.2.2	Modelo Floco de Neve	55
10.2.3	Resumo dos Esquemas Multidimensionais.....	56
11.	DADOS ESTRUTURADOS X DADOS NÃO ESTRUTURADOS	56
12.	OLAP (On-Line Analytical Processing):.....	57
10.3.	Níveis de análise de dados: DESCRITIVO, PREDITIVO e PRESCRITIVO.	58
10.4.	Operações OLAP.....	59
13.	BIG DATA.....	60
13.1.	Objetivo Big Data	61
13.2.	Armazenamento BIG DATA	62
13.2.1	Data Warehouse	62
13.2.2	Data Lake.....	64
13.3.	TIPOS DE ANÁLISE BIG DATA:.....	65
13.4.	5 V's do BIG DATA	67
13.5.1	VOLUME:	67
13.5.2	VELOCIDADE:.....	67
13.5.3	VARIEDADE:.....	67
13.5.4	VERACIDADE.....	68
13.5.5	VALOR	68
13.5.	CLOUD COMPUTING:	70
13.6.	MapReduce	71
13.7.	Hadoop.....	71

13.8.	Normalização Big Data	72
13.9.	PIPELINE Big Data.....	73
13.10.	TIPOS DE VARIÁVEIS.....	73
13.11.1	Variável DEPENDENTE.....	73
13.11.2	Variável INDEPENDENTE	74
13.11.	O gerenciamento de Big Data	74
13.12.	Big Data analítico	75
14.	NoSQL (Not only SQL)	76
14.1.	Teorema CAP (ou Teorema de Brewer)	78
14.2.	Principais Características dos Bancos de Dados NoSQL.....	78
14.3.	Modelo NoSQL Chave-Valor (Key-Value)	83
14.4.	Modelo NoSQL Orientado a Documentos	83
14.5.	Modelo Orientado a Colunas (ou Colunar)	83
14.6.	Modelo NoSQL Orientado a Grafos	84

1. MODELAGEM CONCEITUAL DE DADOS

A arquitetura mais conhecida é a ANSI/SPARC, fundamentada em TRÊS NÍVEIS em que cada um desses níveis corresponde às abstrações dos dados armazenados no banco de dados.

1.1. Níveis de Abstração

A figura seguinte representa esses três níveis de abstração, que são:

- **Nível de Visões do Usuário (Externo);**
- **Nível Lógico (Conceitual);**
- **Nível Físico (Interno).**

(a) Nível de Visões do Usuário (Externo)	<p>É o nível mais alto de abstração, que descreve partes do banco de dados, de acordo com as necessidades de cada usuário, individualmente.</p> <p>Em outras palavras, descreve o modo pelo qual os dados são vistos pelos usuários do sistema gerenciador de banco de dados.</p>
(b) Nível Lógico (Conceitual)	<p>Descreve QUAIS dados estão armazenados e seus relacionamentos.</p> <p>Neste nível, o banco de dados é descrito por meio de estruturas relativamente simples, que podem envolver estruturas complexas no nível físico.</p>
(c) Nível Físico (Interno)	<p>Nível mais baixo de abstração.</p> <p>Descreve COMO os dados estão realmente armazenados, englobando estruturas complexas de baixo nível que são descritas em detalhe.</p>

1.2. Independência de Dados

A independência dos dados nada mais é do que a capacidade de alterar o esquema em um nível dos sistemas de banco de dados sem alterar o esquema no nível **MAIS ALTO** ou, em outras palavras, a habilidade de modificar a definição de um esquema em um nível sem afetar a definição do esquema em um nível mais alto.

É possível definir **dois tipos de independência de dados**:

- **A independência externa** é a capacidade de alterar o esquema externo (ou seja, a maneira como os dados são apresentados aos usuários ou aplicações) sem afetar a maneira como os dados são organizados e armazenados internamente.
- **Independência LÓGICA de dados**: a capacidade de alterar o esquema conceitual sem ter de alterar os esquemas externos ou de programas de aplicação; modifica o esquema lógico sem que, com isso, qualquer programa aplicativo precise ser reescrito. As modificações no nível conceitual são necessárias quando a estrutura lógica do banco de dados é alterada (por exemplo, a adição de contas de bolsas de mercado num sistema bancário);

- **Independência FÍSICA de dados:** a capacidade de alterar o esquema interno sem ter de alterar o esquema conceitual e, por consequência, sem ter que alterar os esquemas externos. modifica o esquema físico sem que, com isso, qualquer programa aplicativo precise ser reescrito (As modificações no nível físico são ocasionalmente necessárias para aumento de desempenho).
 - A criação de índices é uma operação que afeta a organização física dos dados (ou seja, a maneira como eles são armazenados no disco), tornando mais rápida a recuperação de dados específicos, e é feita sem que as aplicações que acessam os dados precisem ser modificadas. Portanto, isso é um exemplo de independência física.
- A **independência conceitual**, por outro lado, é a capacidade de alterar o esquema conceitual sem afetar o esquema lógico. O esquema conceitual é uma visão de alto nível de todo o banco de dados, geralmente descrevendo os tipos de dados armazenados e as relações entre eles, mas sem detalhes sobre como os dados estão realmente organizados no nível lógico ou físico. A independência conceitual permitiria, por exemplo, alterar a forma como os dados são modelados no nível mais alto (por exemplo, **mudando de um modelo relacional para um modelo orientado a objetos**) sem afetar o esquema lógico.

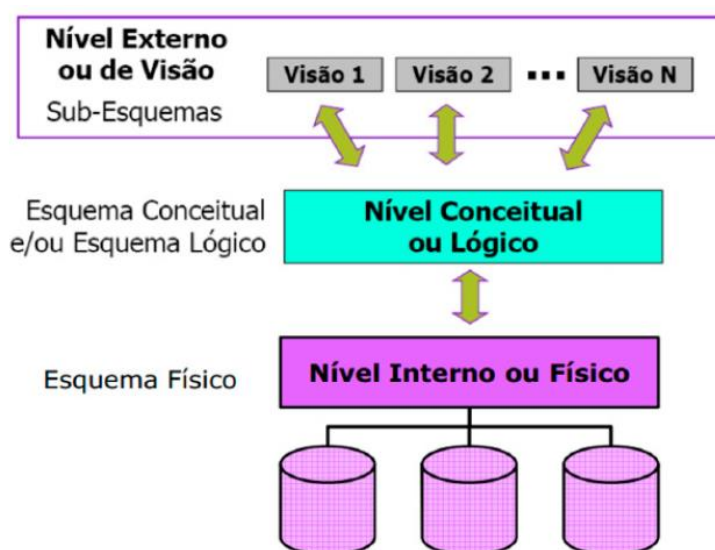
O conceito de independência dos dados é **similar** em muitos aspectos ao conceito de **tipos abstratos de dados** em modernas linguagens de programação. Ambos escondem detalhes de implementação do usuário. Isto permite ao usuário concentrar-se na estrutura geral em vez de detalhes de baixo nível de implementação.

1.3. Arquitetura de Três Níveis

A arquitetura mais conhecida é a ANSI/SPARC, fundamentada em **TRÊS NÍVEIS** em que cada um desses níveis corresponde às abstrações dos dados armazenados no banco de dados.

A figura seguinte representa esses **três níveis de abstração**, que são:

- **Nível de Visões do Usuário (Externo);**
- **Nível Lógico (Conceitual);**
- **Nível Físico (Interno).**



● **Nível externo ou visão** – (também conhecido como nível lógico do usuário) é o mais próximo dos usuários – ou seja, é aquele que se ocupa do modo como os dados são vistos por usuários individuais.

● **Nível conceitual** – (também conhecido como nível lógico de comunidade, ou às vezes apenas nível lógico, sem qualificação) é um nível "indireto" entre os outros dois.

● **Nível interno** – (também conhecido como nível de armazenamento) é o mais próximo do meio de armazenamento físico – ou seja, é aquele que se ocupa do modo como os dados são fisicamente armazenados dentro do sistema.

2. MODELO ENTIDADE RELACIONAMENTO (MODELO CONCEITUAL)

Existem diferentes estratégias para a modelagem do banco dados:

Bottom-Up	Inicia a partir dos conceitos mais detalhados percorrendo o sistema até os mais abstratos.
Top-Down	Ao contrário, parte dos conceitos mais abstratos até os mais detalhados.
Inside-Out	Uma abordagem alternativa é a Inside-out , em que se parte dos conceitos considerados mais relevantes e, gradativamente, se vai adicionando conceitos secundários. Em outras palavras, essa técnica de modelagem E-R inicia nos conceitos mais importantes e navega em direção aos menos importantes . Nessa estratégia, identifica-se o núcleo do sistema e, gradativamente, acrescentam-se novas entidades.

2.1. Atributos

Atributos Simples (ou Atômicos)	Atributos Compostos
Atributos que não são divisíveis (não é dividido em partes). São chamados também por atributos atômicos .	Podem ser divididos em partes menores , ou subpartes, os quais representariam atributos básicos mais simples com significados independentes.
Exemplo: CEP.	Exemplo: o atributo Endereço pode ser subdividido em número, logradouro, cidade, estado e CEP.

Atributos Monovalorados (ou atômicos)	Atributos Multivalorados
Possuem apenas um valor para uma entidade em particular.	Podem assumir múltiplos valores. Uma única entidade tem diversos valores para este atributo. Esse tipo de atributo é <u>representado por uma elipse com linha dupla</u> .

Por exemplo, o atributo CPF de uma entidade Funcionário é monovalorado, pois cada funcionário possui apenas um CPF.

Ex¹: O atributo **telefone** é multivalorado, pois um funcionário pode possuir vários telefones ao mesmo tempo ou até mesmo nenhum valor.

Ex²: O atributo **idioma** de uma entidade aluno pode conter os valores inglês e francês. Para outro aluno poderia conter apenas um valor - espanhol. Para um terceiro aluno, poderíamos ter 3 valores para este atributo.

Atributos Armazenados	Atributos Derivados
Atributos que realmente pretendemos guardar no Banco de Dados .	Podem ser gerados ou calculados a partir de outros atributos.

Por exemplo, o atributo Data de Nascimento de uma entidade Funcionário é armazenado.

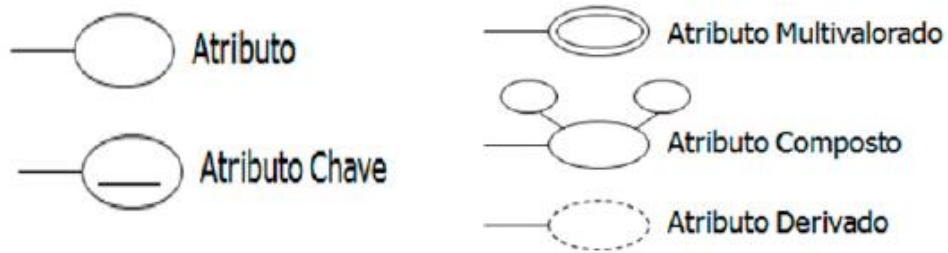
Caso tivéssemos no nosso modelo um atributo armazenado Data de Nascimento, na entidade Funcionário poderíamos ter um atributo derivado idade, calculada a partir da Data de Nascimento.

Atributo Chave (ou Determinante): existem alguns atributos cujos valores **são distintos para cada elemento do conjunto**. Esses atributos são chamados de chaves. Uma chave é um atributo **ÚNICO** para cada elemento do conjunto, servindo para identificar univocamente um elemento.

- O atributo chave identifica (determina) cada elemento de uma Entidade de forma única dentro do conjunto-entidade.



Veja a seguir as formas mais usadas para representação dos atributos no DER.



Descritivos	Descrevem, representam características de um objeto. Ex: Altura, peso, data de nascimento.
Nominativos	Além de descrever, também <u>definem nomes</u> ou rótulos de identificação dos objetos aos quais pertencem. Ex: código, matrícula, número.
Referenciais	Faz referência a outra entidade, como por exemplo, o código do produto em uma nota fiscal.

Cardinalidade de Atributos

Um atributo pode possuir uma cardinalidade, de maneira análoga a uma entidade num relacionamento e essa cardinalidade irá definir quantos valores desse atributo podem estar associados com uma ocorrência da entidade ou relacionamento ao qual ele pertence.

Alguns exemplos:

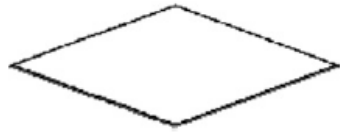
- Cardinalidade (1,1): obrigatória, mas não precisa ser representada no diagrama;
- Cardinalidade (0,1): opcional;
- Cardinalidade (0,n): opcional e multivalorada;



2.2. Relacionamentos

Normalmente, um relacionamento é representado por um **losango** com um verbo para indicar a ação de relacionamento.

NOTAÇÃO:



2.1.1 GRAU DE RELACIONAMENTO:

A primeira característica de um relacionamento é que este tem um **grau**. E o que é o **grau de um relacionamento**? É simplesmente o número de entidades que fazem parte desse relacionamento.

Inicialmente, entendam também grau do relacionamento como cardinalidade do mesmo.

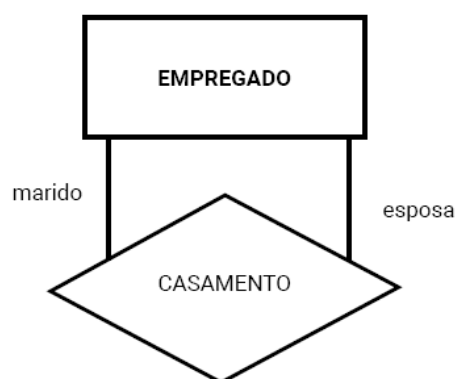
Várias são as possibilidades de relacionamentos.

Uma entidade pode participar de relacionamentos com quaisquer outras entidades do modelo, inclusive com ela mesma.

- a) **Relacionamento Unário (ou Autorrelacionamento)**: quando uma entidade se relaciona com si própria, temos o relacionamento unário, ou autorrelacionamento.

EXEMPLO:

Esse é caso do relacionamento "casamento" da figura seguinte. Neste exemplo estamos lidando com a modelagem de uma empresa em que marido e mulher são empregados desta empresa. Deste modo, na tabela EMPREGADO teríamos a informação do cônjuge. Se ambos são empregados então há um autorrelacionamento.



Relacionamento recursivo, autorrelacionamento e relacionamento unário são sinônimos!

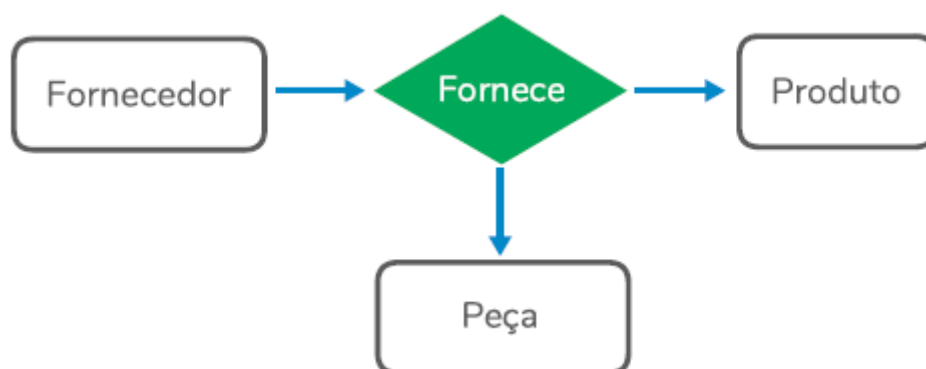
Um exemplo clássico na literatura e praticado em projetos reais no mercado é a entidade FUNCIONÁRIO em que um funcionário pode assumir o papel de gerente, coordenador ou supervisor. Isso quer dizer que um gerente além de ser um simples funcionário também gerencia outros.

b) **Relacionamento Binário:**



- c) **Relacionamentos N-ários:** acima de duas entidades, esses relacionamentos normalmente são denominados n-ários (Podemos ter relacionamentos ternários - com três entidades, quaternário - com quatro entidades, e assim por diante).

A seguir, tem-se um relacionamento ternário:



2.1.2 CARDINALIDADE DE RELACIONAMENTO:

Para definir o número de ocorrências de uma entidade usamos o conceito de **Cardinalidade**, que indica quantas ocorrências de uma entidade participam **no mínimo e no máximo do relacionamento**. Em outras palavras, a cardinalidade de um relacionamento expressa quantas entidades de um grupo se relacionam com uma entidade do outro.

Cardinalidade Mínima: define se o relacionamento entre duas entidades é obrigatório ou não. É o número mínimo de instâncias da entidade associada que devem se relacionar com uma instância da entidade em questão.

Usada para indicar o tipo de participação da entidade em um relacionamento. Esta participação pode ser: parcial/opcional ou total/obrigatória.

a) **Parcial ou opcional:**

- Uma ocorrência da entidade pode ou não participar de determinado relacionamento;
- É indicado pela cardinalidade = 0 (zero);
- A cardinalidade mínima 0 recebe a denominação de associação opcional, uma vez que indica que o relacionamento PODE ou não associar uma ocorrência de entidade a cada ocorrência da outra entidade em questão.



Um Departamento pode ter no mínimo nenhum empregado (0) e, no máximo, vários empregados.

Indica que podem existir departamentos que não tem nenhum empregado relacionado a ele.

(A leitura é sempre feita olhando o lado inverso!)

b) Total ou Obrigatória:

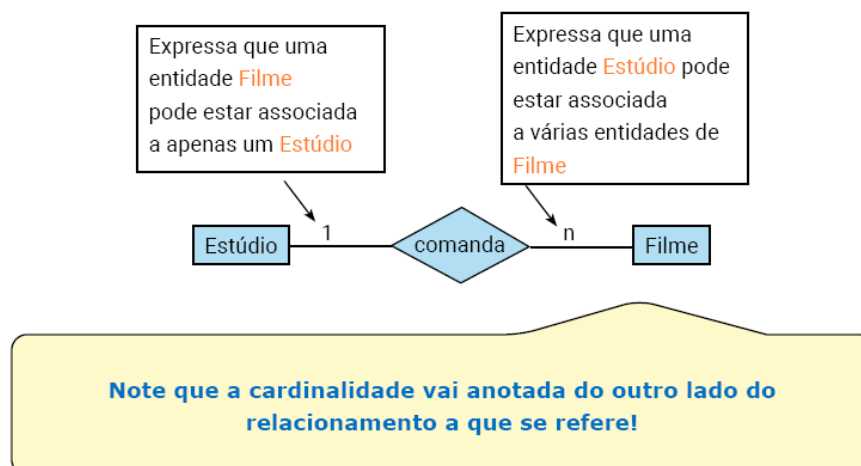
- Quando TODAS as ocorrências de uma entidade devem participar de determinado relacionamento;
- É indicado pela cardinalidade mínima > 0 (zero)... geralmente 1;
- A cardinalidade mínima 1 recebe a denominação de associação obrigatória, uma vez que indica que o relacionamento DEVE obrigatoriamente associar uma ocorrência de entidade a cada ocorrência da outra entidade em questão.



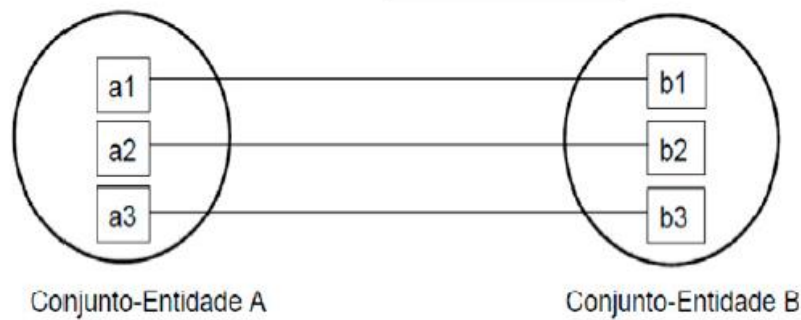
Todos os departamentos devem **possuir pelo menos (no mínimo)** um empregado alocado.

Indica que não poderá existir no banco um departamento que não tenha nenhum empregado.

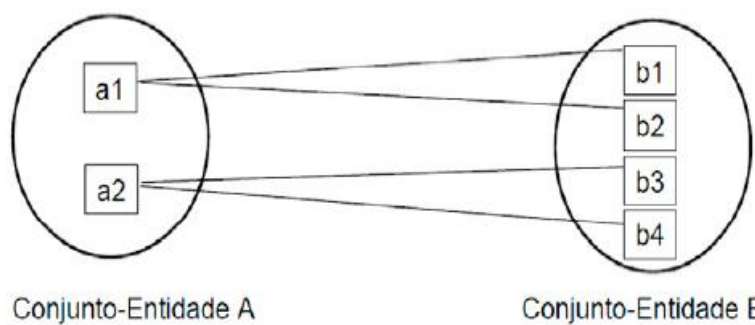
A leitura é sempre feita olhando o lado inverso! Assim, esse exemplo indica que um DEPARTAMENTO deve ter, pelo menos, um empregado ou muitos (1, N) e um EMPREGADO pode não estar associado a nenhum departamento a princípio (0,1). Entretanto, se o empregado estiver alocado em algum departamento este poderá ser em apenas um.



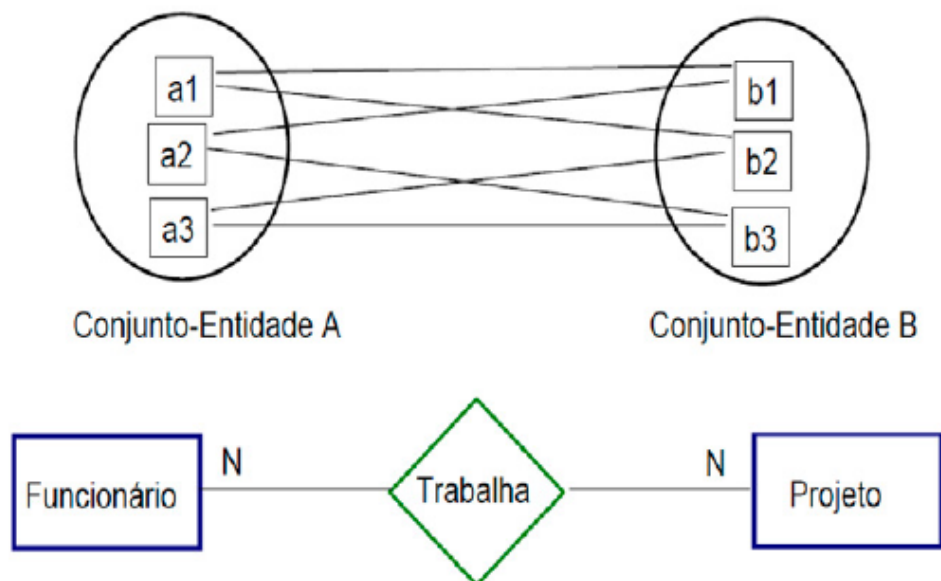
Um-para-um: uma entidade em A está associada no máximo a uma entidade em B e uma entidade em B está associada no máximo a uma entidade em A.



Um-para-muitos: uma entidade em A está associada a qualquer número de entidades em B, enquanto uma entidade em B está associada no máximo a uma entidade em A.



Muitos-para-muitos: uma entidade em A está associada a qualquer número de entidades em B, e uma entidade em B está associada a qualquer número de entidades em A.



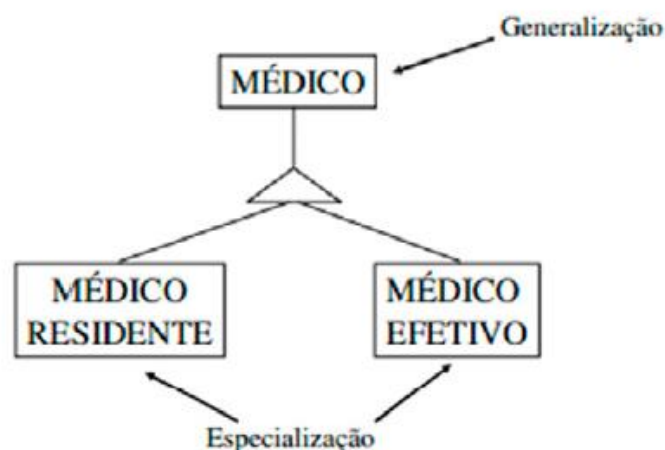
O número de entidades que participam de um tipo relacionamento é irrestrito e armazenam muito mais informações do que diversos relacionamentos binários.

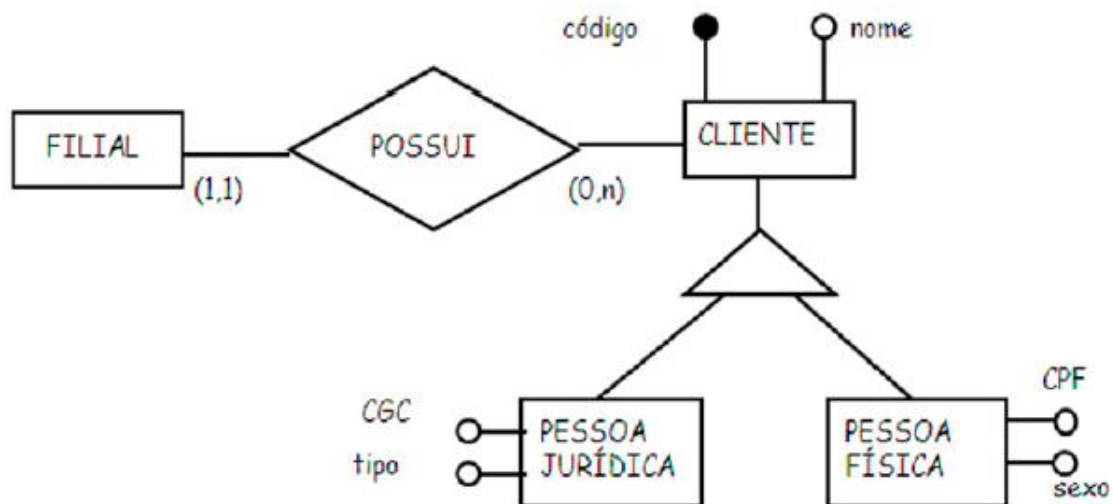
2.3. Entidades

2.3.1 Generalização/Especialização

É possível incluir nos modelos entidade-relacionamento (MER) conjuntos de entidades com diversas características em comum, diferenciando apenas em algumas características.

Nesse caso, pode-se usar o conceito de **generalização**, em que se cria um conjunto de entidades genérico contendo as características em comum, e de **especialização** em que se especializam (apresentam diferenças) nas características que são distintas.





O triângulo representa uma associação de **especialização/ generalização** (Significa que a entidade Cliente foi especializada em duas outras entidades, Pessoa Física e Pessoa Jurídica).

A entidade especializada herda as propriedades da entidade genérica.

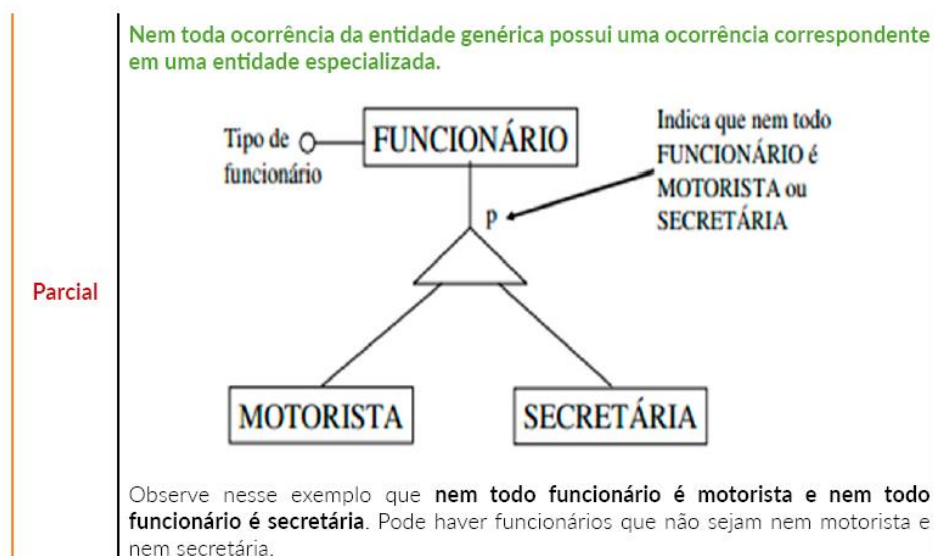
Conforme visto, existem atributos comuns a todos os Clientes, como *código* e *nome*. Se o Cliente for Pessoa Física, ele tem também os atributos *CPF* e *sexo*. Se for Pessoa Jurídica, ele tem também *CGC* e *tipo*.

Veja que os atributos das entidades que especializaram Cliente são os atributos herdados de Cliente (Código e Nome) e mais os atributos próprios.

Assim, **especialização** é o processo de definir um conjunto de subclasses de um tipo de entidade. A entidade que foi especializada é denominada superclasse.

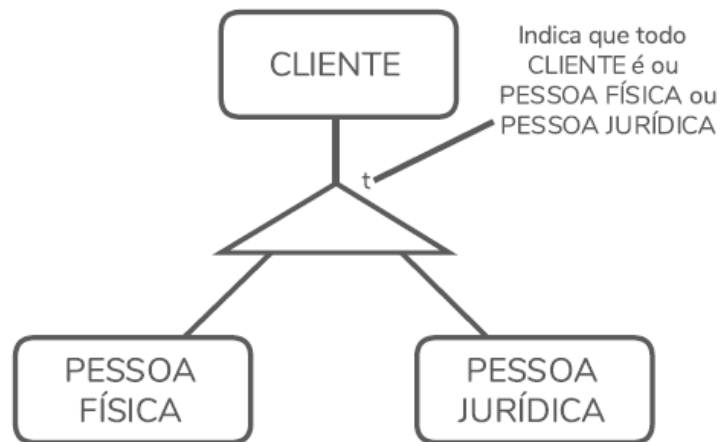
Já a **generalização** é o **processo contrário**, no qual encontramos características comuns de algumas entidades, e criamos uma superclasse para elas.

É como se tivéssemos pensando o modelo anterior com as entidades Pessoa Física e Pessoa Jurídica, e, depois, chegássemos à conclusão de que tudo é cliente, o que caberia generalizar essas duas entidades em uma entidade Cliente, que teria os atributos comuns.



Para toda ocorrência da entidade genérica existe sempre uma ocorrência em uma das entidades especializadas.

Total



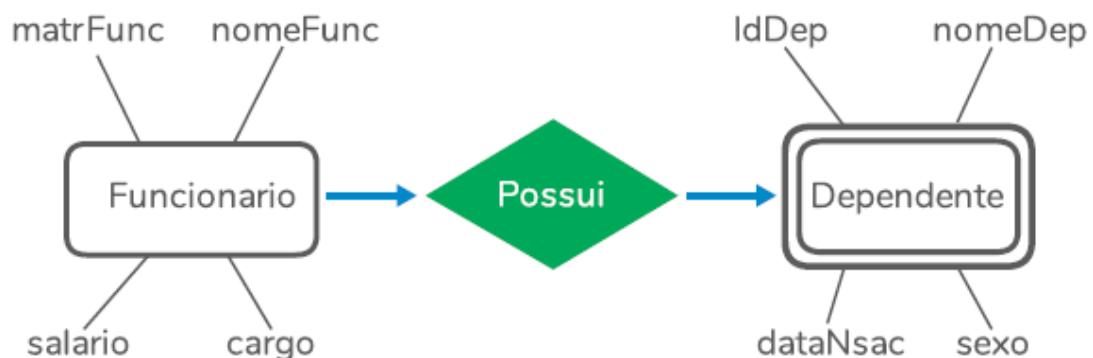
No exemplo acima, todo cliente ou é uma pessoa física ou uma pessoa jurídica. Não existe a possibilidade de haver um cliente que não seja pessoa física OU pessoa jurídica.

2.3.2 Entidade Fraca

Uma **entidade fraca** é uma entidade que tem uma relação de dependência com outra entidade. Isto quer dizer que **a entidade fraca só existe se existir a entidade com a qual está relacionada.**

Um exemplo de entidade fraca é a entidade Dependente, na relação com a entidade Funcionário. Cabe destacar que só existe Dependente se existir Funcionário.

As entidades fracas são representadas por retângulos duplos, e seus relacionamentos também podem ser representados por losangos duplos.



Em entidades fracas, sempre teremos uma participação total da entidade fraca com sua entidade normal, pois **um elemento de uma entidade fraca tem que estar associado a, pelo menos, um elemento da entidade normal.**

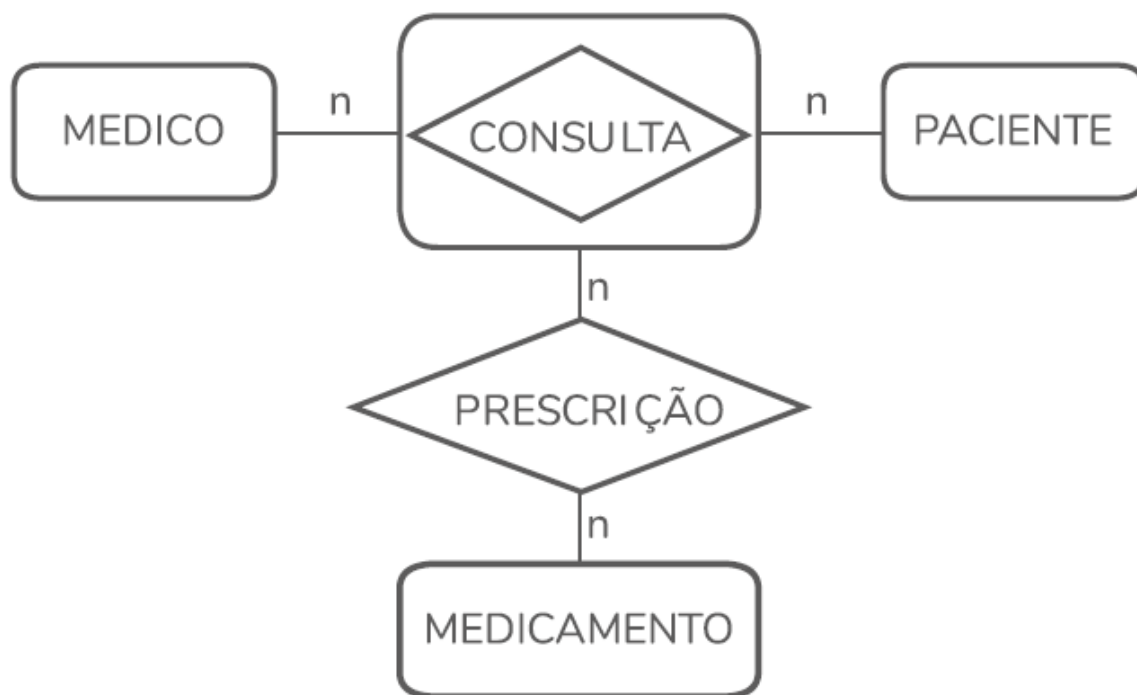
A entidade fraca é aquela que não possui atributos “capazes” de formar uma chave primária. A entidade fraca precisa se relacionar com outra entidade para existir. Afinal, ela não possui chave primária!

2.3.3 Entidade Associativa

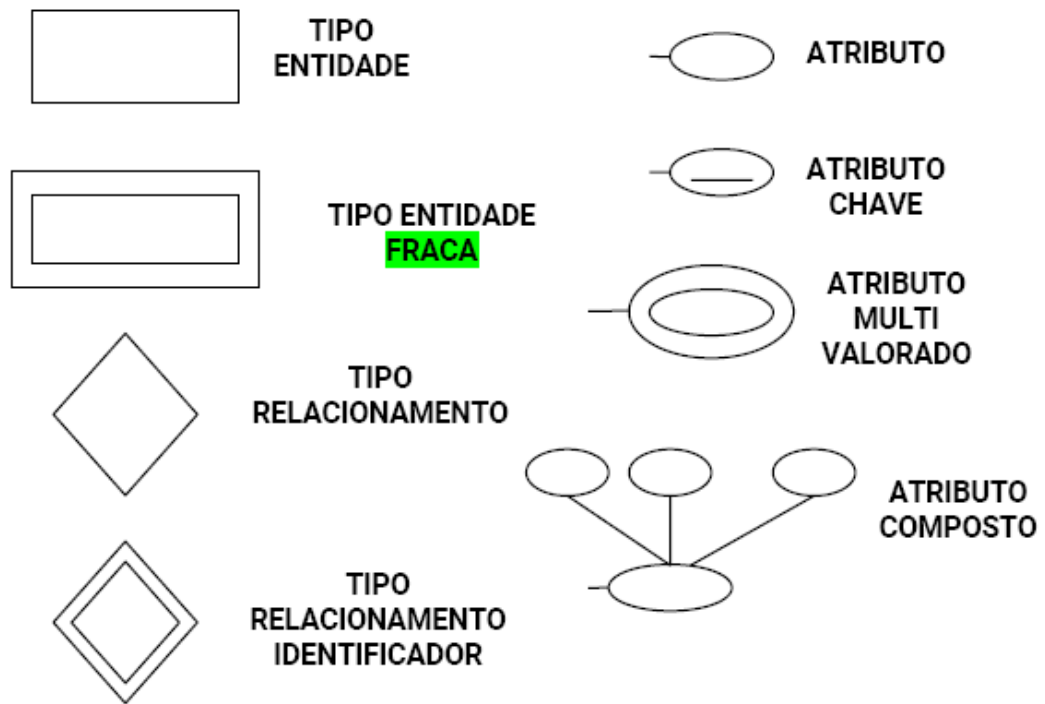
Por definição um **relacionamento** é uma associação entre entidades. Na modelagem ER **não** é prevista a possibilidade de associar uma entidade a um relacionamento, ou de associar dois relacionamentos entre si. Mas, em certas oportunidades, durante a modelagem surgem situações nas quais é desejável permitir uma associação entre uma entidade e um relacionamento, gerando aí uma **Entidade Associativa**.

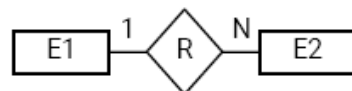
Observe como exemplo o modelo ilustrado a seguir:

Deseja-se modelar a prescrição de medicamentos receitados aos pacientes, com a criação da entidade Medicamentos. A solução é então transformar o relacionamento entre Médico e Paciente numa **Entidade Associativa** e relacioná-la com a entidade Medicamento.

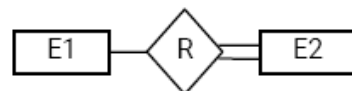


2.4. Diagrama Entidade - Relacionamento (DER)

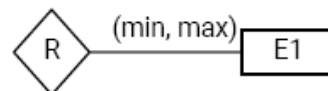




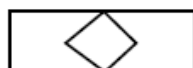
Taxa de Cardinalidade 1:N
para E1:E2 em R



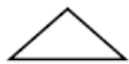
Participação Parcial de E1 em
R, Participação Total de E2 em
R



Restrição Estrutural (min,max)
na Participação de E1 em R



ENTIDADE ASSOCIATIVA



ESPECIALIZAÇÃO



DEPENDÊNCIA

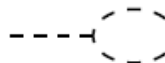


CARDINALIDADE

(MIN. MAX)










ATRIBUTO-CHAVE



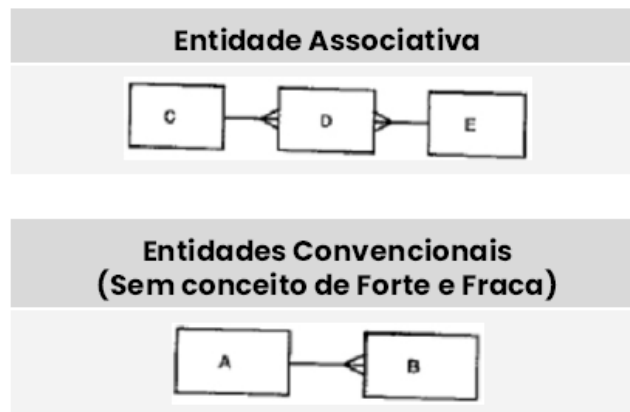
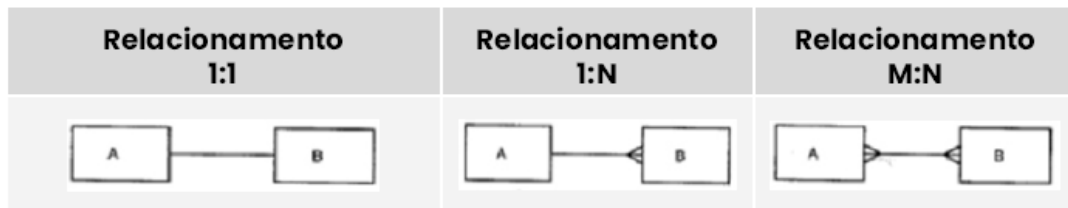
ATRIBUTO DERIVADO

2.5. Notação James Martin (ou Pé de Galinha – Crow's Foot)

Sintaxe	Significado
	Entidade (Aluno) Atributos (CPF, Nome e Endereço), sendo CPF um atributo identificador (ou chave).
	Cardinalidade 1
	Cardinalidade Muitos
	Cardinalidade 0:1 (0,1)
	Cardinalidade 1:1 (1,1)

Sintaxe	Significado
	Cardinalidade 0:N (0,N)
	Cardinalidade 1:N (1,N)

Exemplos:



Structured Query Language (SQL)

3. SGBD (Conceitos)



3.1. BD Operacional x Data Warehouse:

- Banco de dados operacional (transacional ou de produção): Um banco projetado principalmente para dar suporte às operações diárias de uma empresa é classificado como.
- *Data Warehouses* (Armazém de Dados): focam na armazenagem dos dados utilizados para gerar informações necessárias à tomada de decisões táticas e estratégicas. A maioria dos dados de suporte a decisões baseiam-se em dados históricos obtidos de bancos de dados operacionais. Além disso, o Data Warehouse pode armazenar dados provenientes de muitas fontes. Para facilitar a recuperação desses dados, a estrutura do data warehouse difere muito de um banco operacional ou transacional.

Um *Data Warehouse* (armazém de dados, ou depósito de dados), é um repositório de informações colhidas de várias origens, armazenadas sob um esquema unificado, em um único local, que propõe sustentar a tomada de decisão com dados.

Assim, uma das características fundamentais de um Data Warehouse está em proporcionar um ambiente que permita realizar análise dos negócios de uma empresa com base nos dados por ela armazenados.

Para que serve? Para criar uma visão única e centralizada dos dados que estavam dispersos em diversos Bancos de Dados. Permite que usuários finais executem consultas, gerem relatórios e façam análises.

3.2. Sistemas de Gerenciamento de Banco de Dados (SGBDs)

Um SGBD é um SOFTWARE (conjunto de programas) de caráter geral, que executa os processos de definição, construção, manipulação e compartilhamento de bancos de dados entre vários usuários e aplicações, incluindo módulos para consulta, atualização e as interfaces entre o sistema e o usuário.

SGBDs sozinhos não têm nenhuma relevância, os bancos de dados armazenados em um SGBD é que têm significado e são importantes para a organização que os mantém.

Características de um SGBD:

- REDUZIR Redundância de dados significa dados repetidos sem necessidade. E isso não é bom para o SGBD, pois dados repetidos significa espaço sendo desperdiçado. Por isso, os SGBDs tratam os dados de forma a reduzir a redundância o máximo possível. Reduzir redundância pode significar economia financeira, pois há uma menor necessidade de disco rígido.
- Durabilidade: significa que os dados são íntegros, isto é, se um campo tem o valor "Belo Horizonte", e nenhum aplicativo o altera, então este valor será mantido até que algum aplicativo o delete ou o altere.
- Segurança: significa que os dados só serão acessados por usuários que, em algum momento, ganharam permissão para tal. Um usuário em um SGBD pode ter permissão para ler uma tabela, mas não para inserir dados nela.
- Integridade de dados: significa que os campos terão os dados armazenado de forma consistente nos formatos e restrições que lhe foram atribuídos. Por exemplo, se um campo só pode inserir dados maiores do que zero, então um valor negativo não poderá estar neste

campo. Se um campo permite somente 100 caracteres, então não haverá um valor com mais do que 100 caracteres.

Dentre os SGBDs que estão sendo mais utilizados atualmente tem-se:

- SGBDs livres, como o MySQL, que possui o código fonte aberto, projetado para servir como opção aos SGBDs corporativos proprietários. Apesar de seu crescimento nos últimos anos e de sua grande popularidade, especialmente em aplicativos voltados para a web, ainda não tem uso difundido em grandes empresas, que ainda preferem confiar seus dados a aplicações mais “maduras” e com maior capacidade de suporte. Há outros SGBDs livres que seguem a linha do MySQL, como o PostgreSQL, por exemplo;
- Microsoft Access (Faz parte do pacote Microsoft Office, voltado para bancos de dados pessoais (uso doméstico) e menos robustos (pequenas aplicações de uso não crítico));
- Base (Faz parte do pacote BrOffice/LibreOffice, também mais voltado para uso doméstico);
- SGBDs comerciais e proprietários: para uso corporativo, como o SQL Server e o Oracle, utilizados em projetos mais volumosos que envolvem bancos de dados corporativos (de grandes empresas). Outros SGBDs podem ser destacados, como: SyBase, Adabas, DB2, etc.

SGBDs são softwares, já banco de dados conceitualmente NÃO é um software.

3.3. Esquema X Instância:

- Esquema de um banco de dados é a descrição de um banco de dados que é especificada durante o projeto do banco de dados. Ou seja, é o projeto geral do banco de dados. Geralmente, poucas mudanças ocorrem no esquema do banco de dados.
- Instância do banco de dados são os dados armazenados em um determinado instante do tempo. A instância altera toda vez que uma alteração no banco de dados é feita.

Modelos de SGBD :Modelo Relacional (exemplo)

nome	rua	cidade	nro-conta
Mário	Av. S.Carlos	S.P.	1234
Rui	Rua XV	S.Carlos	1333
Rui	Rua XV	S.Carlos	7556
Silvia	Av.D.Pedro	Itu	5512
Silvia	Av.D.Pedro	Itu	7556

nro-conta	saldo
1234	55,00
1333	600,00
5512	350,00
7556	3.000,00

ESQUEMA = Projeto geral do Banco de Dados -> os esquemas são alterados com pouca frequência.

Instância do Banco de Dados = Conjunto de informações contidas em determinado BD em um dado momento.

Esquema de banco de dados corresponde à definição do tipo em uma linguagem de programação. Uma variável de um dado tipo tem um valor em particular em dado instante. Assim, esse valor corresponde a uma instância do esquema do banco de dados." Portanto, um esquema de banco de dados corresponde às declarações de variável em um programa.

3.4. Metadados de Arquivos

Os metadados fornecem uma descrição das características dos dados e do conjunto de relacionamentos que ligam os dados encontrados no Banco de Dados.

- permitem que o usuário direcione a análise, pelo conhecimento da estrutura dos dados.

Metadados armazenam, estruturam e correlacionam, preferencialmente em um repositório de metadados dotado de um metamodelo para apoiar o controle, a divulgação e o consumo.

Metadados podem possuir a classificação de estrutural ou semântico:

- Metadado estrutural: representa a informação que descreve a organização e estrutura dos dados gravados; por exemplo, informações sobre o formato, os tipos de dados usados e os relacionamentos sintáticos entre eles;
- Metadados semânticos: fornecem informações sobre o significado dos dados disponíveis e seus relacionamentos semânticos; por exemplo, dados que descrevem o conteúdo semântico de um valor de dado (como unidades de medida e escala), ou dados que fornecem informações adicionais sobre sua criação (algoritmo de cálculo ou derivação da fórmula usada), linhagem dos dados (fontes) e qualidade (atualidade e precisão).

Outra classificação, destaca os metadados técnicos ou de negócios:

- Metadados técnicos: descrição dos dados necessários para as diversas ferramentas que precisem armazenar, manipular ou movimentar dados. Essa ferramenta pode se tratar de um banco de dados relacional, ferramentas *Computer Aided Software Engineering*(CASE), ferramentas de pesquisa em banco de dados, ferramentas *On-line Analytical Processing* (OLAP), entre outras;
- Metadados de negócios: descrição de dados necessários pelos usuários de negócios, para entender o contexto do negócio e o significado dos dados. Atualmente existem ferramentas só para efeito de documentação. Quando metadados for rotineiramente usado para gerar regras de negócios executáveis, a definição de metadados será a representação de instruções de regra de negócios de acordo com o esquema de classificação que pode ser transformado em sistemas de informação do negócio.

3.5. Dicionário/Catálogo de Dados

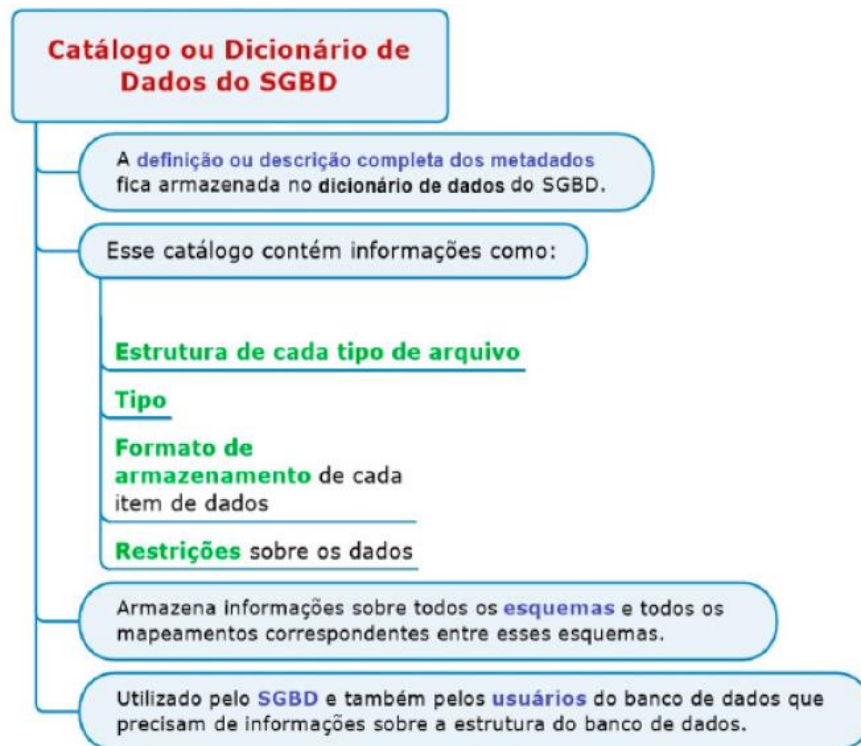
É uma coleção de metadados que contém definições e representações de elementos de dados.

- O dicionário de dados armazena a estrutura de um banco de dados relacional. Ele fornece informações sobre as definições das colunas de uma tabela; define as restrições de integridade e as informações de segurança e domínios definidos pelo usuário.
 - Recurso para garantir a integridade dos dados armazenados em bancos de dados;
- Ponto de partida para o desenvolvimento de um DW é levantar todas as informações que os usuários desejam, fazendo o cruzamento das dimensões e fatos que serão necessários para alcançar o objetivo da organização.

Dentro do contexto de SGBDs, um dicionário de dados é um grupo de tabelas, habilitadas apenas para leitura ou consulta, ou seja, é uma base de dados, propriamente dita, que entre outras coisas, mantém as seguintes informações:

- definição precisa sobre elementos de dados;
- perfis de usuários, papéis e privilégios;
- descrição de objetos;
- integridade de restrições;
- stored procedures e gatilhos;
- estrutura geral da base de dados;
- informação de verificação;
- alocações de espaço.

Um dos benefícios de um dicionário de dados bem preparado é a consistência entre itens de dados por meio de diferentes tabelas. Por exemplo, diversas tabelas podem conter números de telefones. Utilizando uma definição de um dicionário de dados bem-feito, o formato do campo 'número de telefone' definido com "(99)9999-9999" deverá ser obedecido em todas as tabelas que utilizarem esta informação.



4. Linguagem SQL

4.1. Introdução

Como vimos rapidamente, a linguagem Structured Query Language (SQL) é uma linguagem de programação padrão usada para gerenciar e manipular bancos de dados relacionais. É usada para inserir, atualizar e recuperar dados de um banco de dados, bem como gerenciar a estrutura de um banco de dados, incluindo criação de tabelas, índices e restrições. Os comandos SQL mais comuns incluem SELECT, INSERT, UPDATE, DELETE e ALTER.

A linguagem SQL é uma linguagem declarativa, que permite que você acesse e manipule dados armazenados em bancos de dados relacionais. Ela é amplamente utilizada para gerenciar bancos de dados de grande porte, como Oracle Database, MySQL, Microsoft SQL Server e PostgreSQL.

Uma linguagem declarativa é um estilo de programação em que você descreve o resultado desejado, em vez de especificar os passos exatos para alcançá-lo. Em uma linguagem declarativa, você faz declarações sobre o que deseja que aconteça, deixando ao sistema o trabalho de decidir como alcançar o resultado.

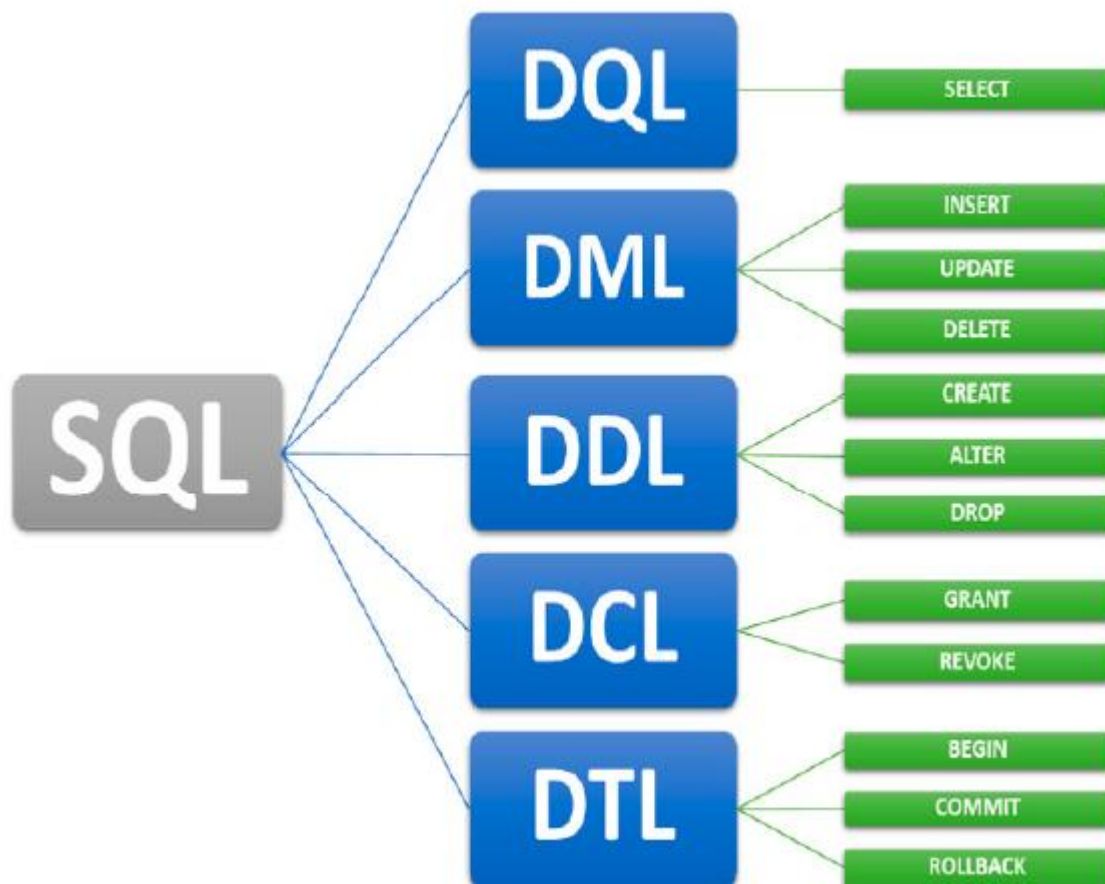
Em comparação, em uma linguagem imperativa, você escreve o código que dita explicitamente as ações a serem executadas para alcançar o resultado desejado. A linguagem SQL é considerada uma linguagem declarativa, porque você escreve declarações sobre os dados que deseja recuperar, em vez de escrever o código para percorrer os dados e selecioná-los manualmente. O sistema de gerenciamento de banco de dados (SGBD) é responsável por decidir como executar a consulta de forma mais eficiente.

4.2. Subconjuntos da Linguagem SQL

A linguagem SQL é composta de vários subconjuntos, cada um com seu próprio conjunto de comandos e funcionalidades. Alguns dos subconjuntos mais comuns incluem:

- **Data Query Language (DQL):** é usado para realizar consultas aos dados armazenados no banco de dados. O comando mais comum é o SELECT.
- **Data Manipulation Language (DML):** é usado para manipular os dados armazenados no banco de dados. Os comandos mais comuns incluem INSERT, UPDATE e DELETE;
- **Data Definition Language (DDL):** é usado para definir a estrutura do banco de dados, incluindo tabelas, colunas, índices e restrições. Os comandos mais comuns incluem CREATE, ALTER e DROP;
- **Data Control Language (DCL):** é usado para controlar o acesso aos dados armazenados no banco de dados. Os comandos mais comuns incluem GRANT e REVOKE;
- **Data Transaction Language (DTL):** é usado para gerenciar transações, garantindo a integridade dos dados e evitando perda de informações. Os comandos mais comuns incluem BEGIN, COMMIT e ROLLBACK;

Cada banco de dados relacional pode implementar uma combinação diferente dos subconjuntos SQL, dependendo de suas necessidades específicas e de sua plataforma.



4.3. Sintaxe Básica

A sintaxe básica da linguagem SQL é composta de comandos que permitem o acesso, a manipulação e o controle de dados armazenados em um banco de dados relacional. Alguns dos principais comandos usados na linguagem SQL incluem:

- **SELECT:** é usado para selecionar dados de uma ou mais tabelas. É possível especificar as colunas que deseja recuperar, bem como filtrar os dados com cláusulas como **WHERE**;
- **FROM:** é um componente obrigatório do comando **SELECT** e indica de qual tabela ou tabelas os dados serão recuperados;
- **WHERE:** é usada para filtrar os dados recuperados pelo comando **SELECT**, baseado em uma ou mais condições. Por exemplo, você pode usar a cláusula **WHERE** para recuperar apenas linhas em que uma coluna específica tenha um valor específico;
- **INSERT:** é usado para inserir novos dados em uma tabela. É possível especificar os valores para cada coluna da tabela;
- **UPDATE:** é usado para atualizar dados existentes em uma tabela. Você pode especificar quais colunas serão atualizadas e os novos valores a serem gravados;
- **DELETE:** é usado para excluir linhas de uma tabela. É possível especificar as linhas que devem ser excluídas, usando a cláusula **WHERE**.

Além dos comandos **SELECT**, **FROM**, **WHERE**, **INSERT**, **UPDATE** e **DELETE**, a sintaxe básica da linguagem SQL inclui outras cláusulas e operadores que ajudam a manipular e organizar dados, em um banco de dados relacional, que também iremos explorar no decorrer da nossa aula. Aqui estão alguns exemplos:

- **JOIN:** essa cláusula é usada para combinar linhas de duas ou mais tabelas com base em uma coluna comum. Isso permite que você recupere dados relacionados de várias tabelas em uma única consulta;
- **GROUP BY:** essa cláusula é usada para agrupar linhas de uma tabela com base em uma ou mais colunas. Isso permite que você obtenha informações agregadas, como somas, médias, contagens etc.;
- **HAVING:** essa cláusula é usada para filtrar os resultados de uma consulta agrupada. Por exemplo, você pode usar a cláusula **HAVING** para recuperar apenas grupos que atendam a determinadas condições;
- **ORDER BY:** essa cláusula é usada para ordenar os resultados de uma consulta. Você pode especificar a ordem crescente ou decrescente e as colunas de acordo com as quais os dados devem ser classificados;

- **DISTINCT**: esse operador é usado para remover linhas duplicadas dos resultados de uma consulta;
- **UNION**: esse operador é usado para combinar os resultados de duas ou mais consultas em uma única tabela resultante;
- **LIKE**: esse operador é usado para recuperar linhas que correspondam a um padrão específico. Por exemplo, você pode usar o operador LIKE para recuperar todas as linhas que contêm determinada string.

Outros comandos importantes da linguagem SQL são CREATE, ALTER e DROP, que são usados para criar, modificar e excluir estruturas de banco de dados, respectivamente. Aqui está um resumo de cada comando:

- **CREATE**: é usado para criar tabelas, índices, visões, procedimentos armazenados, gatilhos etc. em um banco de dados. Ele permite especificar as colunas da tabela, o tipo de dados de cada coluna, as restrições de integridade etc.;
- **ALTER**: é usado para modificar estruturas existentes em um banco de dados, como adicionar ou excluir colunas de uma tabela, modificar o tipo de dados de uma coluna, adicionar ou excluir restrições etc.;
- **DROP**: é usado para excluir estruturas existentes em um banco de dados, como tabelas, índices, visões etc. É importante usar esse comando com cuidado, pois ele remove permanentemente as estruturas e todos os dados armazenados nelas.

4.4. A instrução Select:

- a) Para selecionar todas as colunas de uma tabela:

```
SELECT * FROM tabela;
```

- b) Para selecionar apenas algumas colunas de uma tabela:

```
SELECT coluna1, coluna 2, coluna 3 FROM tabela;
```

- c) Para selecionar dados com uma condição WHERE:

```
SELECT coluna1, coluna 2 FROM tabela
WHERE coluna3 = 'valor';
```

- d) Para selecionar dados com várias condições WHERE:

```
SELECT coluna1, coluna2 FROM tabela
WHERE coluna3 = 'valor1'
AND coluna4 = 'valor2';
```

- e) Evitar informações repetidas e redundantes.

```
SELECT DISTINCT * FROM tabela;
```

4.5. As Instruções JOIN

As instruções JOIN são usadas em consultas para combinar informações de duas ou mais tabelas em uma única tabela resultante. As tabelas são relacionadas com base em uma ou mais colunas comuns.

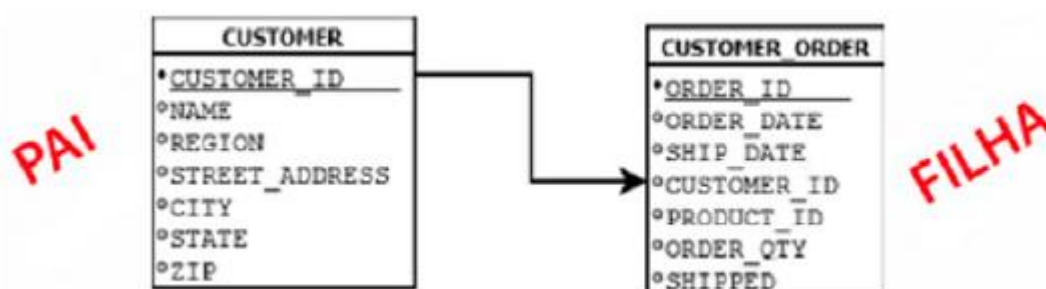
Por exemplo, considere a tabela `customer_order`, que tem um campo `customer_id`:

	ORDER ID	ORDER DATE	SHIP DATE	CUSTOMER ID	PRODUCT ID	ORDER QTY	SHIPPED
1	3	2015-04-20	2015-04-23	3	5	300	false
2	4	2015-04-18	2015-04-22	5	4	375	false
3	1	2015-04-15	2015-04-18	1	1	450	false
4	5	2015-04-17	2015-04-20	3	2	500	false
5	2	2015-04-18	2015-04-21	3	2	600	false

O campo `customer_id` fornece uma chave para buscas na tabela `customer`. Dessa forma, a tabela `customer` também possui um campo `customer_id`:

CUSTOMER ID	NAME	REGION	STREET ADDRESS	CITY	STATE	ZIP
1	LITE Industrial	Southwest	729 Ravine Way	Irving	TX	75014
2	Rex Tooling Inc	Southwest	6129 Collie Blvd	Dallas	TX	75201
3	Re-Barre Construction	Southwest	9043 Windy Dr	Irving	TX	75032
4	Prairie Construction	Southwest	264 Long Rd	Moore	OK	62104
5	Marsh Lane Metal Works	Southeast	9143 Marsh Ln	Avondale	LA	79782

Esse é um exemplo de relacionamento entre a tabela `customer_order` e a tabela `customer`. Podemos dizer que `customer` é pai de `customer_order`, uma vez que `customer_order` depende das informações de `customer`, logo ela é filha de `customer`. Inversamente, `customer` não pode ser filha de `customer_order`, porque não depende dela para obter informações. O diagrama abaixo demonstra esse relacionamento:



A seta mostra que `customer` fornece informações para `customer_order` via `customer_id`.

Os relacionamentos entre tabelas são a maneira como o banco de dados define a ligação entre duas ou mais tabelas. Esses relacionamentos são criados usando chaves primárias e estrangeiras.

A chave primária é um campo único, em uma tabela, que identifica de forma exclusiva cada registro na tabela. No nosso exemplo, a tabela `customer` possui o campo `customer_id` como chave primária.

Já a chave estrangeira é usada para criar uma ligação entre duas tabelas. A chave estrangeira pode fazer referência a uma chave primária ou qualquer outra chave candidata de outra tabela e permite que os dados de uma tabela sejam relacionados com os dados de outra tabela. No nosso, a tabela

customer_order possui a chave estrangeira customer_id, que faz referência à chave primária customer_id na tabela customer.

Esses relacionamentos permitem que as informações de uma tabela sejam combinadas com as informações de outra tabela, o que é útil para realizar consultas mais complexas ou para garantir a integridade dos dados.

Outro aspecto que devemos considerar, em um relacionamento, é quantos registros da tabela-filha podem estar associados a um único registro da tabela-pai.

Analisando as tabelas do nosso exemplo (customer e customer_order), podemos observar que se trata de um relacionamento um-para-muitos (1:N), em que um único registro de cliente pode estar associado a vários pedidos. Observe, na figura abaixo, o cliente "Re-Barre Construction", que tem a ID de cliente 3 e está associado a três pedidos:

CUSTOMER ID	NAME	REGION	STREET ADDRESS	CITY	STATE	ZIP
1	LITE Industrial	Southwest	729 Ravine Way	Irving	TX	75014
2	Rex Tooling Inc	Southwest	6129 Collie Blvd	Dallas	TX	75201
3	Re-Barre Construction	Southwest	9043 Windy Dr	Irving	TX	75032
4	Prairie Construction	Southwest	264 Long Rd	Moore	OK	62104
5	Marsh Lane Metal Works	Southeast	9143 Marsh Ln	Avondale	LA	79782

ORDER_ID	ORDER_DATE	SHIP_DATE	CUSTOMER_ID	PRODUCT_ID	ORDER_QTY	SHIPPED
1	3 2015-04-20	2015-04-21	3	5	300	false
2	4 2015-04-18	2015-04-22	5	4	375	false
3	1 2015-04-15	2015-04-18	1	1	450	false
4	5 2015-04-17	2015-04-20	3	2	500	false
5	2 2015-04-18	2015-04-21	3	2	600	false

No relacionamento um-para-muitos (1:N), uma linha de uma tabela (tabela "A") pode ser relacionada a múltiplas linhas em outra tabela (tabela "B"), mas cada linha da tabela "B" só pode ser relacionada a uma linha na tabela "A".

Além do relacionamento um-para-muitos, há outros três tipos de relacionamentos comuns entre tabelas em um banco de dados.

No relacionamento muitos-para-muitos (N:M), uma linha em uma tabela pode ser relacionada a múltiplas linhas em outra tabela e vice-versa.

EXEMPLO

Uma tabela de produtos pode ser relacionada a uma tabela de categorias, em que cada produto pode pertencer a várias categorias, e cada categoria pode conter muitos produtos.

No relacionamento um-para-um (1:1), uma linha em uma tabela só pode ser relacionada a uma única linha em outra tabela e vice-versa.

EXEMPLO

Uma tabela de funcionários pode ser relacionada a uma tabela de detalhes do funcionário, em que cada funcionário tem seus próprios detalhes, e cada detalhe corresponde a um único funcionário.

No relacionamento auto-relacionamento, uma tabela é relacionada consigo mesma.

EXEMPLO

Uma tabela de funcionários pode ser relacionada consigo mesma para representar relações hierárquicas, em que cada funcionário tem um gerente, e o gerente é um funcionário.

O INNER JOIN retorna somente as linhas que têm correspondentes em ambas as tabelas e permite mesclar duas tabelas. Porém, para mesclar tabelas, precisamos definir um atributo comum entre elas, para que seus registros se alinhem. Precisamos definir um ou mais campos que elas tenham em comum e fazer a associação a partir deles. Se quisermos consultar a tabela customer_order e associá-la a customer, para acessar informações de clientes, é preciso definir customer_id como atributo comum.

Observe um exemplo de operação de INNER JOIN:

```
SELECT order_id,  
customer.customer_id,  
order_date,  
ship_date,  
name,  
street_address,  
city,  
state,  
zip,  
product_id,  
order_qty  
FROM customer INNER JOIN customer_order  
ON customer.customer_id = customer_order.customer_id;
```

Esse comando realiza um INNER JOIN entre as tabelas customer e customer_order. O INNER JOIN retorna apenas as linhas em que há correspondência entre as chaves especificadas na cláusula ON.

Na cláusula ON, a coluna customer_id, na tabela customer, é comparada com a coluna customer_id na tabela customer_order. Quando as colunas coincidem, as linhas correspondentes são combinadas e retornadas como uma única linha na consulta. A saída será uma tabela que combina as informações de ambas as tabelas, exibindo os detalhes do cliente em cada pedido.



	ORDER ID	CUSTOMER ID	ORDER DATE	SHIP DATE	NAME	STREET ADDRESS	CITY	STATE	ZIP	PRODUCT ID	ORDER QTY
1	1	1	2015-05-15	2015-05-18	LITE Industrial	729 Ravine Way	Irving	TX	75014	1	450
2	2	3	2015-05-18	2015-05-21	Re-Barn Construction	9043 Windy Dr	Irving	TX	75032	2	600
3	3	3	2015-05-20	2015-05-23	Re-Barn Construction	9043 Windy Dr	Irving	TX	75032	5	300
4	4	5	2015-05-18	2015-05-22	Marsh Lane Metal Works	9143 Marsh Ln	Aurora	LA	79782	4	375
5	5	3	2015-05-17	2015-05-20	Re-Barn Construction	9043 Windy Dr	Irving	TX	75032	2	500

A operação de INNER JOIN só irá exibir os registros que existam nas duas tabelas.

CUSTOMER

CUSTOMER ID	NAME	REGION	STREET ADDRESS	CITY	STATE	ZIP
1	LITE Industrial	Southwest	729 Ravine Way	Irving	TX	75014
2	Rex Tooling Inc.	Southwest	6129 Collie Blvd	Dallas	TX	75201
3	Re-Barre Construction	Southwest	9043 Windy Dr	Irving	TX	75032
4	Prairie Construction	Southwest	264 Long Rd	Moore	OK	73104
5	Marsh Lane Metal Works	Southeast	9143 Marsh Ln	Avondale	LA	79782

CUSTOMER_ORDER

ORDER ID	ORDER DATE	SHIP DATE	CUSTOMER ID	PRODUCT ID	ORDER QTY	SHIPPED
1	2015-05-15	2015-05-18	1	1	450	false
2	2015-05-18	2015-05-21	3	2	600	false
3	2015-05-20	2015-05-23	3	5	300	false
4	2015-05-18	2015-05-22	5	4	375	false
5	2015-05-17	2015-05-20	3	2	500	false

INNER JOINED

ORDER ID	CUSTOMER ID	ORDER DATE	SHIP DATE	NAME	STREET ADDRESS	CITY	STATE	ZIP	PRODUCT ID	ORDER QTY
1	1	2015-05-15	2015-05-18	LITE Industrial	729 Ravine Way	Irving	TX	75014	1	450
2	3	2015-05-18	2015-05-21	Re-Barre Construction	9043 Windy Dr	Irving	TX	75032	2	600
3	3	2015-05-20	2015-05-23	Re-Barre Construction	9043 Windy Dr	Irving	TX	75032	5	300
4	5	2015-05-18	2015-05-22	Marsh Lane Metal Works	9143 Marsh Ln	Avondale	LA	79782	4	375
5	3	2015-05-17	2015-05-20	Re-Barre Construction	9043 Windy Dr	Irving	TX	75032	2	500

Com o uso de INNER JOIN, qualquer registro que não tenha um valor comum nas duas tabelas será excluído. Se quisermos incluir todos os registros da tabela customer, podemos fazê-lo com LEFT JOIN.

O LEFT JOIN, ou LEFT OUTER JOIN, retorna todas as linhas da tabela esquerda (primeira tabela na consulta) e as correspondentes na tabela direita. Se não houver correspondente na tabela direita, os valores serão nulos.

Vamos refazer a consulta anterior, só que agora substituindo INNER JOIN por LEFT JOIN.

A tabela especificada no lado “esquerdo” do operador LEFT JOIN (customer) terá todos os seus registros incluídos, mesmo se não tiverem registros que possuam algum tipo de relacionamento com os registros da tabela da “direita” (customer_order).

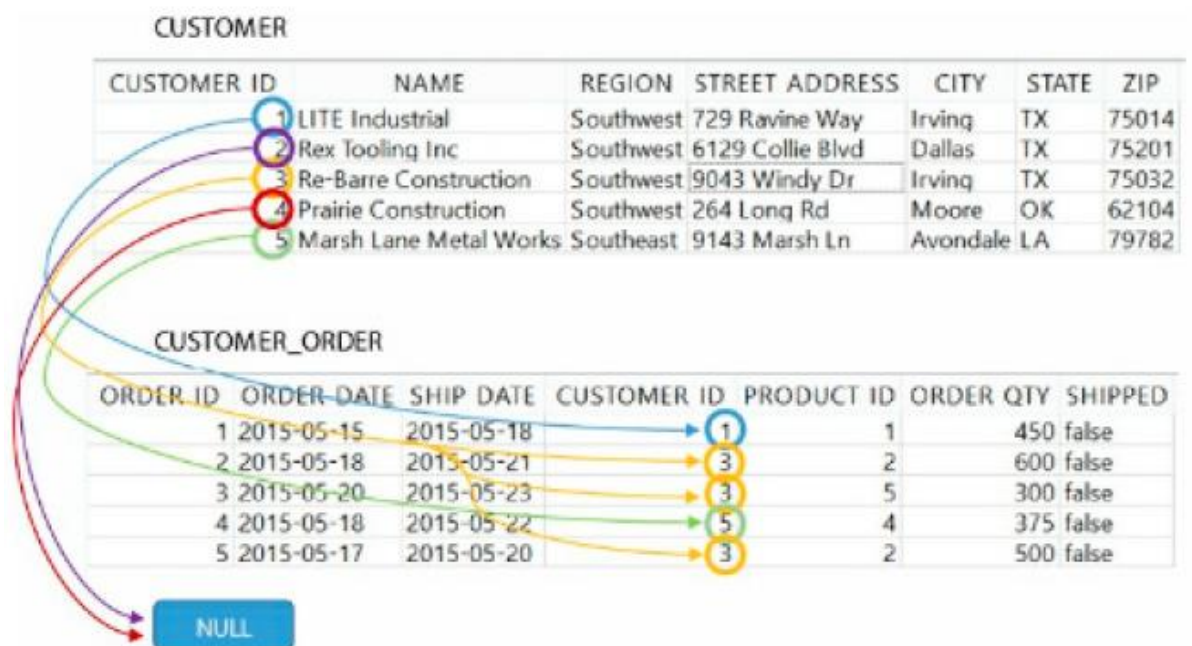
```
SELECT CUSTOMER.CUSTOMER_ID,
NAME,
STREET_ADDRESS,
CITY,
STATE,
ZIP,
ORDER_DATE,
SHIP_DATE,
ORDER_ID,
PRODUCT_ID,
ORDER_QTY
FROM CUSTOMER LEFT JOIN CUSTOMER_ORDER
ON CUSTOMER.CUSTOMER_ID = CUSTOMER_ORDER.CUSTOMER_ID
```

Tabela da “esquerda” Tabela da “direita”

Ao executar essa operação, teremos resultados semelhantes aos obtidos na consulta INNER JOIN anterior, mas serão acrescentados os registros que existem na tabela customer, mesmo que não possuam relacionamento com registros da tabela customer_order.

CUSTOMER ID	NAME	STREET ADDRESS	CITY	STATE	ZIP	ORDER DATE	SHIP DATE	ORDER ID	PRODUCT ID	ORDER QTY
1	LITE Industrial	729 Ravine Way	Irving	TX	75014	2015-05-15	2015-05-18	1	1	450
2	Rex Tooling Inc	6129 Collie Blvd	Dallas	TX	75201	NULL	NULL	NULL	NULL	NULL
3	Re-Barre Construction	9043 Windy Dr	Irving	TX	75032	2015-05-17	2015-05-20	5	2	500
3	Re-Barre Construction	9043 Windy Dr	Irving	TX	75032	2015-05-18	2015-05-21	2	2	600
3	Re-Barre Construction	9043 Windy Dr	Irving	TX	75032	2015-05-20	2015-05-23	3	5	300
4	Prairie Construction	264 Long Rd	Moore	OK	62104	NULL	NULL	NULL	NULL	NULL
5	Marsh Lane Metal Works	9143 Marsh Ln	Avondale	LA	79782	2015-05-18	2015-05-22	4	4	375

Observe que, nos registros que não possuem relacionamento da tabela customer, todos os campos que são oriundos da tabela customer_order são nulos, porque não há pedidos a serem incluídos na associação.



LEFT OUTER JOINED

CUSTOMER ID	NAME	STREET ADDRESS	CITY	STATE	ZIP	ORDER DATE	SHIP DATE	ORDER ID	PRODUCT ID	ORDER QTY
1	LITE Industrial	729 Ravine Way	Irving	TX	75014	2015-05-15	2015-05-18	1	1	450
2	Rex Tooling Inc	6129 Collie Blvd	Dallas	TX	75201	NULL	NULL	NULL	NULL	NULL
3	Re-Barre Construction	9043 Windy Dr	Irving	TX	75032	2015-05-17	2015-05-20	5	2	500
3	Re-Barre Construction	9043 Windy Dr	Irving	TX	75032	2015-05-18	2015-05-21	2	2	600
3	Re-Barre Construction	9043 Windy Dr	Irving	TX	75032	2015-05-20	2015-05-23	3	5	300
4	Prairie Construction	264 Long Rd	Moore	OK	62104	NULL	NULL	NULL	NULL	NULL
5	Marsh Lane Metal Works	9143 Marsh Ln	Avondale	LA	79782	2015-05-18	2015-05-22	4	4	375

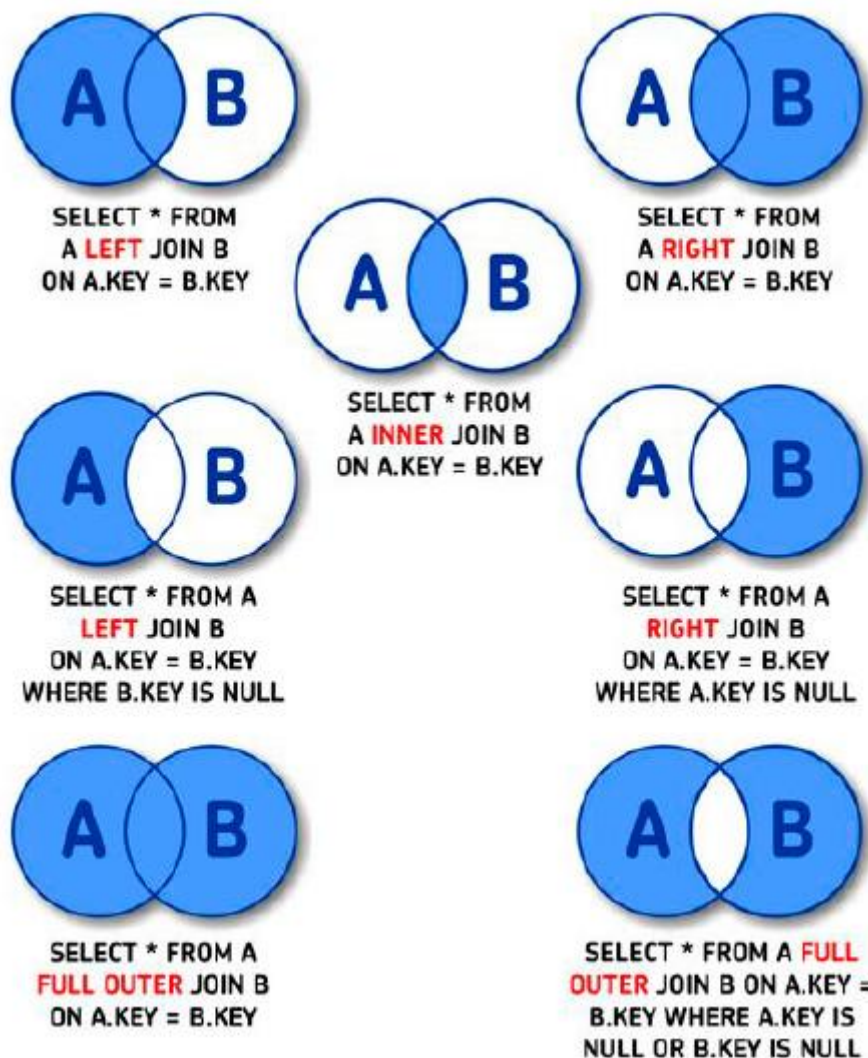
O RIGHT JOIN, ou RIGHT OUTER JOIN, retorna todas as linhas na tabela direita (segunda tabela na consulta) e as correspondentes na tabela esquerda. As linhas na tabela à esquerda, que não têm correspondência na tabela à direita, serão retornadas com valores nulos para as colunas da tabela à direita. O RIGHT JOIN executa uma associação externa à direita, que é quase idêntica à associação externa à esquerda. Ele inverte a direção da associação e inclui todos os registros da tabela da direita.

A sintaxe é semelhante ao LEFT JOIN, exceto que LEFT JOIN é substituído por RIGHT JOIN. A cláusula ON especifica a condição de junção, que determina como as linhas de ambas as tabelas são combinadas. As colunas selecionadas na cláusula SELECT são aquelas retornadas na consulta resultante.

Também há um operador de associação externa completa, chamado de FULL OUTER JOIN. O FULL OUTER JOIN é um tipo de JOIN que combina todas as linhas de duas tabelas, independentemente de haver ou não correspondência entre elas. Se houver uma correspondência, as colunas combinadas são exibidas na saída. Se não houver correspondência, as colunas das tabelas que não correspondem são preenchidas com valores nulos.

O FULL OUTER JOIN é útil quando você quer obter uma visão completa dos dados em ambas as tabelas, incluindo linhas que não correspondem a uma correspondência na outra tabela. Por exemplo, se você tiver uma tabela de clientes e outra tabela de pedidos, o FULL OUTER JOIN permitiria que você tivesse uma visão de todos os clientes, independentemente se eles fizeram algum pedido ou não.

Observe o infográfico abaixo a respeito do uso das instruções JOIN na linguagem SQL:



4.6. A Instrução CREATE

A instrução CREATE é usada para criar um objeto em um banco de dados, como uma tabela, uma visão, um índice, um procedimento armazenado, entre outros. É a primeira etapa na criação de objetos, no banco de dados, e é usada para especificar suas colunas, tipos de dados, restrições, chaves primárias, entre outros aspectos importantes.

4.6.1 Create Table

Por exemplo, para criar uma tabela chamada clientes, com colunas para armazenar informações de nome, endereço e data de nascimento, a instrução seria:

```
CREATE TABLE clientes (  
  nome VARCHAR(50),  
  endereco VARCHAR(100),  
  data_nascimento DATE  
);
```

Essa instrução define a estrutura da tabela e as colunas que a compõem, as suas propriedades e os tipos de dados que são aceitos. No nosso exemplo, estamos criando três colunas (nome, endereço e data_nascimento) junto com a tabela clientes, em que:

- nome: coluna de tipo VARCHAR com comprimento máximo de 50 caracteres;
- endereco: coluna de tipo VARCHAR com comprimento máximo de 100 caracteres;
- data_nascimento: coluna de tipo DATE para armazenar datas.

Observe outro exemplo:

```
CREATE TABLE COMPANY (  
  COMPANY_ID INTEGER PRIMARY KEY AUTOINCREMENT,  
  NAME VARCHAR(30) NOT NULL,  
  DESCRIPTION VARCHAR(60),  
  PRIMARY_CONTACT_ID INTEGER NOT NULL  
);
```

Analisando esse comando, temos que a instrução CREATE TABLE declara uma nova tabela chamada COMPANY. Tudo que se encontra em parênteses depois disso define as colunas da tabela. Cada coluna é definida por um nome, seguido por um tipo de dado e por qualquer restrição ou regra existente, como PRIMARY KEY, AUTOINCREMENT ou NOT NULL.

Os tipos de dados são categorias de dados que especificam o tipo de informação que pode ser armazenado em uma coluna de uma tabela. Por exemplo, VARCHAR é um tipo de dado que armazena uma string de caracteres, enquanto INT ou INTEGER é um tipo de dado que armazena números inteiros.

As restrições são utilizadas para limitar o tipo de dados que pode ser inserido em uma tabela e para garantir a integridade dos dados. Algumas restrições comuns incluem PRIMARY KEY, FOREIGN KEY, NOT NULL e UNIQUE. Essas restrições são usadas para garantir a integridade dos dados, impedir entradas duplicadas e manter a consistência entre as tabelas relacionadas.

Observe outros exemplos com a instrução CREATE:

```
CREATE TABLE ROOM (  
  ROOM_ID INTEGER PRIMARY KEY AUTOINCREMENT,
```

```
FLOOR_NUMBER INTEGER NOT NULL,
SEAT_CAPACITY INTEGER NOT NULL
);
```

Esse comando está criando uma tabela chamada ROOM. A tabela tem três colunas:

- ROOM_ID: coluna do tipo inteiro, que é definida como a chave primária da tabela. O modificador AUTOINCREMENT significa que o valor da coluna será incrementado, automaticamente, cada vez que uma nova linha for inserida na tabela;
- FLOOR_NUMBER: coluna do tipo inteiro, que tem a restrição NOT NULL, o que significa que o valor dessa coluna não pode ser nulo;
- SEAT_CAPACITY: coluna do tipo inteiro, que tem a restrição NOT NULL, o que significa que o valor dessa coluna também não pode ser nulo.

```
CREATE TABLE PRESENTATION (
PRESENTATION_ID INTEGER PRIMARY KEY AUTOINCREMENT,
BOOKED_COMPANY_ID INTEGER NOT NULL,
BOOKED_ROOM_ID INTEGER NOT NULL,
START_TIME TIME,
END_TIME TIME
);
```

Esse comando está criando uma tabela chamada PRESENTATION. A tabela tem cinco colunas:

- PRESENTATION_ID: chave primária, identificador único para cada linha na tabela. É definida como um INTEGER e é gerada automaticamente pelo banco de dados (AUTOINCREMENT);
- BOOKED_COMPANY_ID: coluna que armazena o ID da empresa que reservou a apresentação. É definida como um INTEGER e não pode ser nula (NOT NULL);
- BOOKED_ROOM_ID: coluna que armazena o ID da sala que foi reservada para a apresentação. É definida como um INTEGER e não pode ser nula (NOT NULL);
- START_TIME e END_TIME: colunas que armazenam o horário de início e término da apresentação respectivamente. São ambas definidas como um tipo de dados TIME.

```
CREATE TABLE ATTENDEE (
ATTENDEE_ID INTEGER PRIMARY KEY AUTOINCREMENT,
FIRST_NAME VARCHAR(30) NOT NULL,
LAST_NAME VARCHAR(30) NOT NULL,
PHONE INTEGER,
EMAIL VARCHAR(30),
VIP BOOLEAN DEFAULT(0)
);
```

O comando acima está criando uma tabela chamada ATTENDEE e tem os seguintes atributos:

- ATTENDEE_ID: identificador único para cada linha da tabela, sendo definido como uma chave primária autoincrementada (INTEGER PRIMARY KEY AUTOINCREMENT);
- FIRST_NAME: nome do participante da apresentação, sendo definido como uma string com tamanho máximo de 30 caracteres e não podendo ser nulo (VARCHAR (30) NOT NULL);
- LAST_NAME: sobrenome do participante da apresentação, sendo definido como uma string com tamanho máximo de 30 caracteres e não podendo ser nulo (VARCHAR (30) NOT NULL);

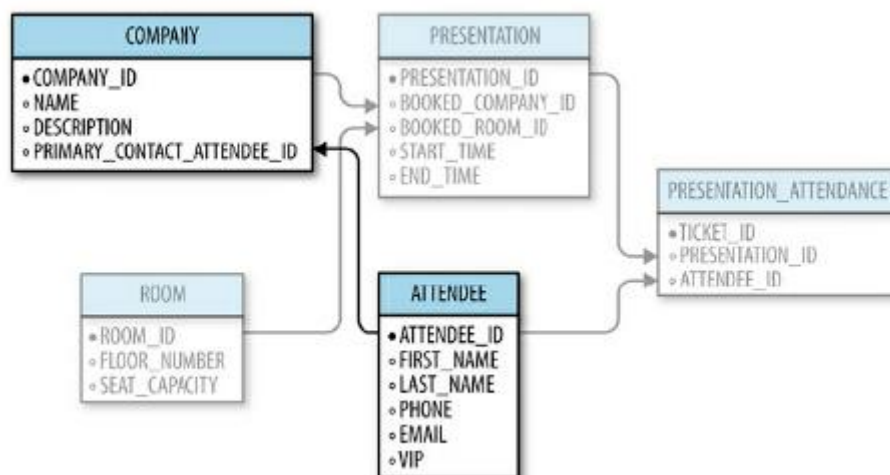
- PHONE: número de telefone do participante da apresentação, sendo definido como um inteiro (INTEGER);
- EMAIL: endereço de e-mail do participante da apresentação, sendo definido como uma string com tamanho máximo de 30 caracteres (VARCHAR (30));
- VIP: indicador booleano que indica se o participante da apresentação é um convidado VIP, sendo definido como um valor-padrão de FALSE (0) (BOOLEAN DEFAULT (0)).

```
CREATE TABLE PRESENTATION_ATTENDANCE (
  TICKET_ID INTEGER PRIMARY KEY AUTOINCREMENT,
  PRESENTATION_ID INTEGER,
  ATTENDEE_ID INTEGER
);
```

O comando acima está criando uma tabela chamada PRESENTATION_ATTENDANCE e tem os seguintes atributos:

- TICKET_ID: chave primária autoincrementada do tipo inteiro;
- PRESENTATION_ID: coluna do tipo inteiro, que representa o ID de uma apresentação;
- ATTENDEE_ID: coluna do tipo inteiro, que representa o ID de um participante.

Observe o relacionamento entre as tabelas criadas:



Observe que, na criação das tabelas, foram definidas apenas as chaves primárias, mas não as chaves estrangeiras, para estabelecer o relacionamento entre as tabelas. Esse relacionamento é criado para garantir integridade de dados e evitar a inserção de dados inválidos na tabela que contém a chave estrangeira. Quando um registro é inserido na tabela que possui a chave estrangeira, o sistema verifica se o valor inserido nesta corresponde a um valor existente na tabela referenciada. Se o valor não existir, a inserção não é permitida, garantindo que apenas dados válidos sejam armazenados. Além disso, se um registro na tabela referenciada for apagado ou modificado, todos os registros correspondentes na tabela com a chave estrangeira serão atualizados ou apagados automaticamente, mantendo a consistência dos dados.

Uma chave estrangeira pode referenciar qualquer chave candidata (chave primária ou chave única) de outra tabela, desde que elas sejam do mesmo tipo de dado.

Por exemplo, analisando as tabelas que criamos, não pode existir um registro na tabela PRESENTATION com um valor para BOOKED_COMPANY_ID que não exista na coluna COMPANY_ID da tabela COMPANY. Se houver um valor igual a 5 para BOOKED_COMPANY_ID, é preciso que também

haja um registro de valor igual a 5 para COMPANY_ID na tabela COMPANY. Caso contrário, haverá um registro órfão. Podemos impor isso, definindo restrições de chave estrangeira.

A chave estrangeira pode ser definida durante a criação da tabela (dentro da instrução CREATE) ou após a tabela já ter sido criada (fazendo uso da instrução ALTER). Por exemplo, para adicionar a restrição de chave estrangeira na tabela PRESENTATION, fazendo referência à tabela COMPANY:

```
ALTER TABLE PRESENTATION
ADD CONSTRAINT FK_BOOKED_COMPANY_ID
FOREIGN KEY (BOOKED_COMPANY_ID)
REFERENCES COMPANY (COMPANY_ID);
```

Esse comando é usado para adicionar uma restrição de chave estrangeira à tabela PRESENTATION. A restrição FK_BOOKED_COMPANY_ID é adicionada na coluna BOOKED_COMPANY_ID da tabela PRESENTATION e faz referência à coluna COMPANY_ID da tabela COMPANY. Isso significa que todo valor na coluna BOOKED_COMPANY_ID da tabela PRESENTATION deve ser igual a um valor existente na coluna COMPANY_ID da tabela COMPANY. Isso é importante para garantir a integridade dos dados e evitar referências inválidas aos dados da tabela COMPANY.

4.6.2 Create View

Outra aplicação para instrução CREATE seria a criação de visões (views). As visões são objetos virtuais, em um banco de dados, que agem como tabelas, mas não armazenam dados. Elas representam uma seleção de dados de uma ou mais tabelas e permitem que um usuário visualize e acesse os dados de forma simplificada. As visões são úteis porque permitem aos desenvolvedores criar um modelo lógico de dados que seja mais fácil de usar e compreender do que a estrutura física subjacente.

- **VIEW:** Visões são tabelas virtuais derivadas de outras tabelas físicas do banco de dados, que podem ser utilizadas para restringir usuários de verem tabelas inteiras.

A criação de uma visão é feita usando a instrução CREATE VIEW, seguida por uma consulta SELECT que define os dados que serão exibidos na visão. Por exemplo:

```
CREATE VIEW view_clientes AS
SELECT nome, endereco, data_nascimento
FROM clients;
```

4.6.3 Create TRIGGER

Dentro do contexto de bancos de dados relacionais SQL, por vezes é necessário realizar uma determinada ação, de acordo com algum evento que acontece dentro do banco de dados. Essa ação pode ser viabilizada através de Triggers (gatilhos). Basicamente, sempre que precisamos disparar uma função através da ocorrência de determinada ação dentro de um banco de dados, isto é, sempre que uma ação ocorre em determinada tabela, podemos incluir um gatilho que ficará responsável por executar determinada tarefa.

De forma geral, as ações que podem realizar o disparo das triggers são especificamente operações de insert (inserção), delete (deleção) e update (atualização). Dessa forma, as triggers geralmente fazem parte de instruções DML (Data Manipulation Language), entretanto podem também ter característica DDL (Data Definition Language) ou de LOGON, sendo acionadas, respectivamente, através de eventos que alteram a estrutura do banco ou quando o usuário está realizando o LOGON no banco de dados. Basicamente, os gatilhos possuem as seguintes vantagens: maior facilidade de

manutenção do sistema, melhor desempenho, facilidade na administração, etc. Por via de regra, a sintaxe básica para criar um gatilho é:

Por via de regra, a sintaxe básica para criar um gatilho é:

```
CREATE TRIGGER [NOME_DA_TRIGGER]
ON [NOME_DA_TABELA]
[BEFORE/FOR/AFTER/INSTEAD OF] [INSERT/UPDATE/DELETE]
AS
CORPO_DO_TRIGGER
```

Esses parâmetros principais, basicamente se referem a:

- **BEFORE/FOR/AFTER/INSTEAD OF** - Grupo de parâmetros que definem o momento que o gatilho é disparado. Nem todos estão disponíveis em todos os SGBDs.
 - **BEFORE** faz o gatilho ser disparado antes da ação.
 - **FOR** é padrão e geralmente faz o gatilho juntamente com a ação.
 - **AFTER** faz o gatilho disparar somente após a ação ativadora ser concluída.
 - **INSTEAD OF** faz o gatilho ser executado no lugar da ação que o ativou.

4.7. A Instrução INSERT

Uma vez que temos nossas tabelas e demais objetos criados, partimos para a manipulação de quais dados serão inseridos, atualizados ou excluídos.

Em um banco de dados relacional só existem dados quando o banco de dados recebe registros. A instrução INSERT faz exatamente isso, insere registros no banco de dados. A instrução INSERT é usada para adicionar linhas (registros) em uma tabela no banco de dados.

Você pode inserir valores em todas as colunas da tabela ou apenas em algumas colunas específicas; nesse caso, as outras colunas serão preenchidas com valores padrão ou valores nulos, dependendo da definição da tabela.

É importante observar que o tipo de dados dos valores inseridos deve ser compatível com o tipo de dados das colunas na tabela. Além disso, as restrições de atributo também devem ser observadas ao inserir dados na tabela.

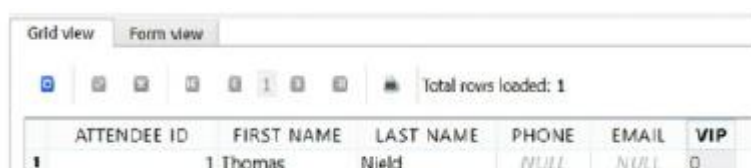
Por exemplo:

```
INSERT INTO ATTENDEE (FIRST_NAME, LAST_NAME)
VALUES ('Thomas', 'Nield');
```

Explicando de forma detalhada o comando acima, a instrução INSERT INTO é seguida pelo nome da tabela ATTENDEE, que será alvo da operação de inserção. Em seguida, entre parênteses, estão listados os nomes das colunas que serão preenchidas com os dados. A instrução VALUES é seguida por uma lista de valores a serem inseridos na tabela, na ordem correspondente à lista de colunas fornecida. Nesse caso, serão inseridos os valores 'Thomas' e 'Nield' na coluna FIRST_NAME e LAST_NAME respectivamente.

Para verificar se o valor foi inserido corretamente, podemos executar um SELECT na tabela:

```
SELECT * FROM ATTENDEE;
```



The screenshot shows a database application interface with two tabs: 'Grid view' and 'Form view'. The 'Grid view' is active, displaying a table with the following data:

ATTENDEE ID	FIRST NAME	LAST NAME	PHONE	EMAIL	VIP
1	Thomas	Nield	NULL	NULL	0

At the top right of the grid, it says 'Total rows loaded: 1'.

Podemos observar vários pontos aqui. Primeiramente, não preenchemos todas as colunas durante o INSERT, mas, de acordo com as regras definidas na criação da tabela ATENDEE, algumas das colunas receberam um valor-padrão ou um valor nulo.

O campo ATENDEE_ID se autoatribuiu o valor 1, devido às regras de PRIMARY KEY e AUTOINCREMENT. Se você inserisse outro registro, ele receberia automaticamente uma ATENDEE_ID igual a 2, depois um valor 3, e assim por diante. Como foi definida uma regra AUTOINCREMENT, você deve evitar preencher o campo ATENDEE_ID por conta e deixar que o próprio sistema de gerenciamento de banco de dados (SGBD) atribua o valor de ID.

Os atributos PHONE e EMAIL não foram especificados durante o INSERT, dessa forma receberam valores nulos. Se uma dessas colunas tivesse uma restrição NOT NULL, sem uma política de valor-padrão, o INSERT teria falhado.

O status VIP também não foi especificado na instrução INSERT, entretanto especificamos a esse campo um valor-padrão igual a falso (0). Portanto, em vez de receber um valor nulo, o sistema de gerenciamento de banco de dados (SGBD) recorreu ao uso do valor-padrão especificado.

Caso você se depare com muitos registros para serem inseridos, não é preciso executar uma instrução INSERT para cada um deles; podemos especificar vários registros a serem inseridos em um único comando INSERT. Para isso, precisamos separar com uma vírgula cada registro que será inserido após a instrução VALUES.

Por exemplo:

```
INSERT INTO ATENDEE (FIRST_NAME, LAST_NAME, PHONE, EMAIL, VIP)
VALUES
('Jon', 'Skeeter', 4802185842, 'john.skeeter@rex.net', 1),
('Sam', 'Scala', 2156783401, 'sam.scala@gmail.com', 0),
('Brittany', 'Fisher', 5932857296, 'brittany.fisher@outlook.com', 1);
```

Também é possível inserir registros usando os resultados de uma consulta SELECT. É necessário que o resultado da instrução SELECT respeite a ordem de inserção das colunas específicas na instrução INSERT, além de possuir os mesmos tipos de dados:

```
INSERT INTO ATENDEE (FIRST_NAME, LAST_NAME, PHONE, EMAIL)
SELECT FIRST_NAME, LAST_NAME, PHONE, EMAIL
FROM SOME_OTHER_TABLE;
```

É possível fazer INSERT sem especificar o nome das colunas, porém todos os dados têm que se passados e na ordem correta:

EMPREGADO

MATRICULA	NOME	DATA_NASC	CERT_RESRV
11111	Paulo Menezes	24/05/1991 00:00	234811
22222	Ana Maria Carvalho	25/07/1983 00:00	null
33333	Alexandre Cardoso	11/08/1989 00:00	101678

```
INSERT INTO EMPREGADO VALUES(55555,'Antônia Pinto',datetime('1994-04-01'),NULL); - RESPOSTA CORRETA -
```


O comando não passa de forma explícita os nomes das colunas, dessa forma os dados são informados na ordem correta. A coluna CERT_RESRV da tabela EMPREGADO recebe um dado *NULL* através desse comando, o que não gera qualquer problema, pois a coluna pode receber dados *NULL*, como é possível ver no exemplo da tabela passada pela questão.

INSERT INTO EMPREGADO VALUES(66666,'Adriana Andrade',datetime('1985-06-04')); - **RESPOSTA ERRADA** - O comando não passa de forma explícita os nomes das colunas, por isso os dados são informados na ordem correta, entretanto a coluna CERT_RESRV não recebe dados o que irá gerar um erro na execução do comando. Ainda que a coluna aceite dados nulos, alguma informação precisa ser indicada.

4.8. A Instrução UPDATE

A instrução UPDATE é usada para atualizar registros existentes em uma tabela do banco de dados. Junto com a instrução UPDATE, é preciso usar a palavra-chave SET, que é seguida pelo nome da coluna e pelo novo valor que você deseja atribuir a ela. Para atualizar os valores de EMAIL, por exemplo, com o intuito de que todos os valores para EMAIL fiquem em maiúsculas, poderíamos fazer isso utilizando a instrução UPDATE junto com a função UPPER():

```
UPDATE ATENDEE SET EMAIL = UPPER (EMAIL);
```

Também podemos atualizar vários campos ao mesmo tempo. Basta separar cada expressão posterior à palavra-chave SET com uma vírgula. Para atualizar tanto o campo FIRST_NAME quanto o campo LAST_NAME para maiúsculas, execute este comando:

```
UPDATE ATENDEE SET  
FIRST_NAME = UPPER (FIRST_NAME),  
LAST_NAME = UPPER (LAST_NAME);
```

Esse comando irá alterar os valores de FIRST_NAME e LAST_NAME para todos os registros existentes na tabela ATENDEE, visto que não definimos nenhuma condição de filtragem.

Também podemos usar a instrução UPDATE junto com a cláusula WHERE, que especifica quais registros devem ser atualizados, usando uma condição lógica. Se você não incluir a cláusula WHERE, todos os registros da tabela serão atualizados. Por exemplo:

```
UPDATE ATENDEE SET FIRST_NAME = 'John'  
WHERE ATENDEE_ID = 1;
```

No comando acima será atualizado apenas o registro na tabela ATENDEE, em que ATENDEE_ID é igual a 1. A coluna FIRST_NAME será alterada para 'John'. A cláusula WHERE é usada para especificar qual registro deve ser atualizado. O ATENDEE_ID é usado como identificador único para selecionar o registro correto. A atualização será aplicada apenas ao registro que atender ao critério de seleção na cláusula WHERE.

Observe outro exemplo:

```
UPDATE ATENDEE SET VIP = 1  
WHERE ATENDEE_ID IN (3, 4);
```

O comando acima modifica o valor da coluna VIP para 1 para todas as linhas na tabela em que o ATENDEE_ID é igual a 3 ou 4.

4.9. A Instrução DELETE

A instrução DELETE é usada para excluir linhas de uma tabela em um banco de dados. Essa instrução pode ser usada para excluir uma única linha ou várias linhas de uma tabela. Caso não seja utilizada uma condição WHERE para especificar quais registros serão excluídos, a instrução DELETE irá deletar todos os registros da tabela:

```
DELETE FROM ATTENDEE;
```

Para remover todos os registros que não possuem informações de contato, podemos filtrar os registros em que PHONE e EMAIL sejam nulos:

```
DELETE FROM ATTENDEE  
WHERE PHONE IS NULL  
AND EMAIL IS NULL;
```

Outro exemplo:

```
DELETE FROM ATTENDEE  
WHERE ATTENDEE_ID = 1;
```

O comando acima é usado para excluir um ou mais registros da tabela ATTENDEE em que a coluna ATTENDEE_ID seja igual a 1. As linhas excluídas pela instrução DELETE não podem ser recuperadas. Além disso, se houver restrições de chave estrangeira na tabela, pode ser necessário excluir primeiro as linhas relacionadas antes de excluir a linha na tabela principal.

4.10. A Instrução GRANT

As instruções GRANT, REVOKE e DENY são usadas para controlar o acesso a objetos do banco de dados, como tabelas, visões, procedimentos armazenados e outros.

A instrução GRANT é usada para conceder permissões a usuários ou grupos de usuários para acessar e manipular objetos do banco de dados. Aqui está um exemplo de comando GRANT:

```
GRANT SELECT, INSERT ON ATTENDEE TO USER1;
```

Nesse exemplo, estamos concedendo as permissões de SELECT e INSERT na tabela ATTENDEE para o usuário USER1. Isso significa que o usuário USER1 agora pode selecionar dados da tabela ATTENDEE e inserir novos registros na tabela.

4.11. A Instrução REVOKE

A instrução REVOKE é usada para revogar permissões concedidas anteriormente a usuários ou grupos de usuários. Por exemplo:

```
REVOKE SELECT ON ATTENDEE FROM USER1;
```

Nesse exemplo, o usuário USER1 perderá o privilégio de selecionar dados da tabela ATTENDEE.

4.12. Restrições SQL

As principais Restrições usadas no SQL são:

- NOT NULL - Garante que uma coluna não pode ter um valor NULL
- UNIQUE - Garante que todos os valores em uma coluna sejam diferentes
- PRIMARY KEY - Uma combinação de NOT NULL e UNIQUE. Identifica exclusivamente cada linha em uma tabela
- FOREIGN KEY - Identifica exclusivamente uma linha / registro em outra tabela
- CHECK - Garante que todos os valores em uma coluna satisfaçam uma condição específica
- DEFAULT - Define um valor padrão para uma coluna quando nenhum valor é especificado
- INDEX - usado para criar e recuperar dados do banco de dados muito rapidamente

5. DADO, INFORMAÇÃO, CONHECIMENTO E INTELIGÊNCIA

5.1. Dados:

Dados podem ser definidos como sucessões de fatos brutos, que não foram organizados, processados, relacionados, avaliados ou interpretados, representando apenas partes isoladas de eventos, situações ou ocorrências. Constituem as unidades básicas a partir das quais informações poderão ser elaboradas ou obtidas, menor partícula estruturada que compõe uma informação.

Outro ponto importante é que os dados podem ser facilmente armazenados e manipulados por um computador. A análise de dados permite que um conjunto de dados seja melhor entendido e organizado, mas para isso precisamos descrever os dados de forma adequada. Essa descrição é feita quando transformamos os dados em informações. Falaremos sobre esse processo em instantes. Antes, vejamos algumas formas factíveis para definição de dados.

DADOS

- Dados são correspondências de um atributo, característica ou propriedade que, sozinho, não tem significado.
- Dados são elementos brutos, sem significado, desvinculados da realidade.
- Dados são simples observações sobre o estado do mundo.
- Dados são um conjunto de fatos objetivos e discretos sobre eventos.
- Dados são a menor partícula estruturada que compõe uma informação.

5.2. Informação:

DADOS + SIGNIFICADO E RELEVÂNCIA = INFORMAÇÃO

- Aumento na compreensão e diminuição da incerteza.
- Possui propósito e contexto.
- Envolve coleta, organização, categorização, orientação, combinação e interpretação.
- → Resultado do processo de acrescentar significado aos dados.

Quando os dados passam por algum tipo relacionamento, avaliação, interpretação ou organização tem-se a geração de informação. A partir do momento em que dados são transformados em informações, decisões podem ser tomadas.

A informação pode ser definida como um dado acrescido de contexto, relevância e propósito.

A informação é gerada a partir de uma interpretação sobre os dados, estes podem ser contextualizados, categorizados, calculados ou condensados. São fatos sobre uma situação, pessoa ou evento definidos em um determinado contexto. Podemos ainda transformar os dados em informação com significado filtrando, ordenando, estruturando. Percebamos que não existe conclusão, mas a organização dos dados neste momento permite que algum tipo de análise possa ser executado.

INFORMAÇÃO

- Conjunto de dados com significado que reduza a incerteza ou que permita o conhecimento a respeito de algo.
- Conjunto dos dados presentes em um contexto, carregado de significados e entregue à pessoa adequada.
- Conhecimento inscrito (gravado) sob a forma escrita (impressa ou numérica), oral ou audiovisual.
- Conjunto de dados contextualizados que visam fornecer uma solução para determinada situação de decisão.
- Fatos e/ou dados que encontramos nas publicações, na internet ou mesmo aquilo que as pessoas trocam entre si.
- Resultado do processo de acrescentar significado aos dados.
- Dados sobre determinado assunto que possam ser interpretados ou tenham significado para o receptor.

5.3. Conhecimento

INFORMAÇÃO + REFLEXÃO e SÍNTESE = CONHECIMENTO

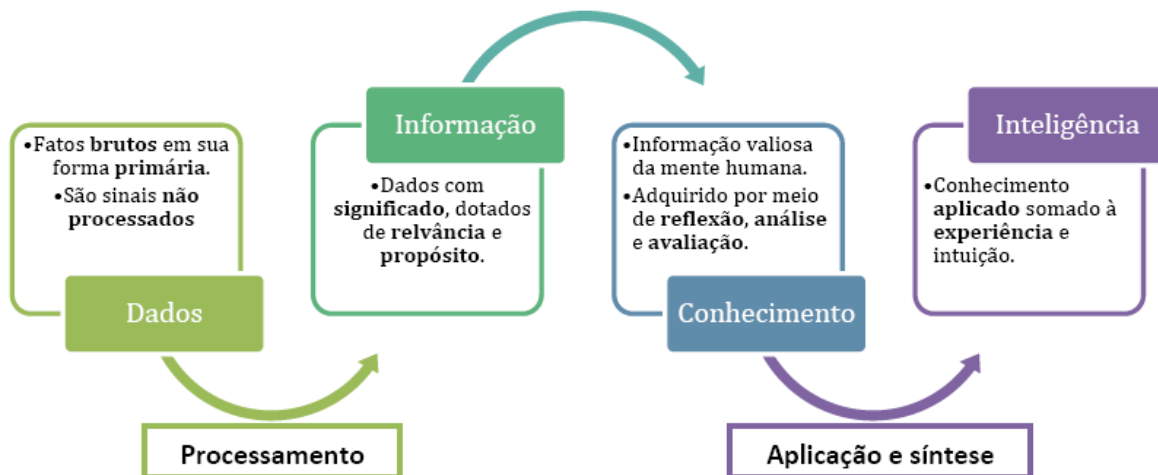
- Informação em ação, contextual, relevante e acionável.
- Informação valiosa da mente humana. Inclui reflexão, síntese e contexto.
- Propriedade subjetiva, inerente a quem analisa os dados ou as informações.
- Difícil de armazenar e estruturar.

CONHECIMENTO

- Propriedade subjetiva, inerente a quem analisa os dados ou as informações.
- Conhecer é o processo de compreender e interiorizar as informações recebidas, possivelmente combinando-as de forma a gerar mais conhecimento.

DADOS	INFORMAÇÃO	CONHECIMENTO
<p>Simple observações sobre o estado do mundo.</p> <ul style="list-style-type: none"> • Facilmente estruturado • Facilmente obtido por máquinas • Frequentemente quantificado • Facilmente transferido 	<p>Dados dotados de relevância e propósito</p> <ul style="list-style-type: none"> • Requer unidade de análise • Exige consenso em relação ao significado • Exige mediação humana 	<p>Informação valiosa da mente humana. Inclui reflexão, síntese e contexto</p> <ul style="list-style-type: none"> • De difícil estruturação • De difícil captura em máquinas • Frequentemente tácito • De difícil transferência.

5.4. Inteligência



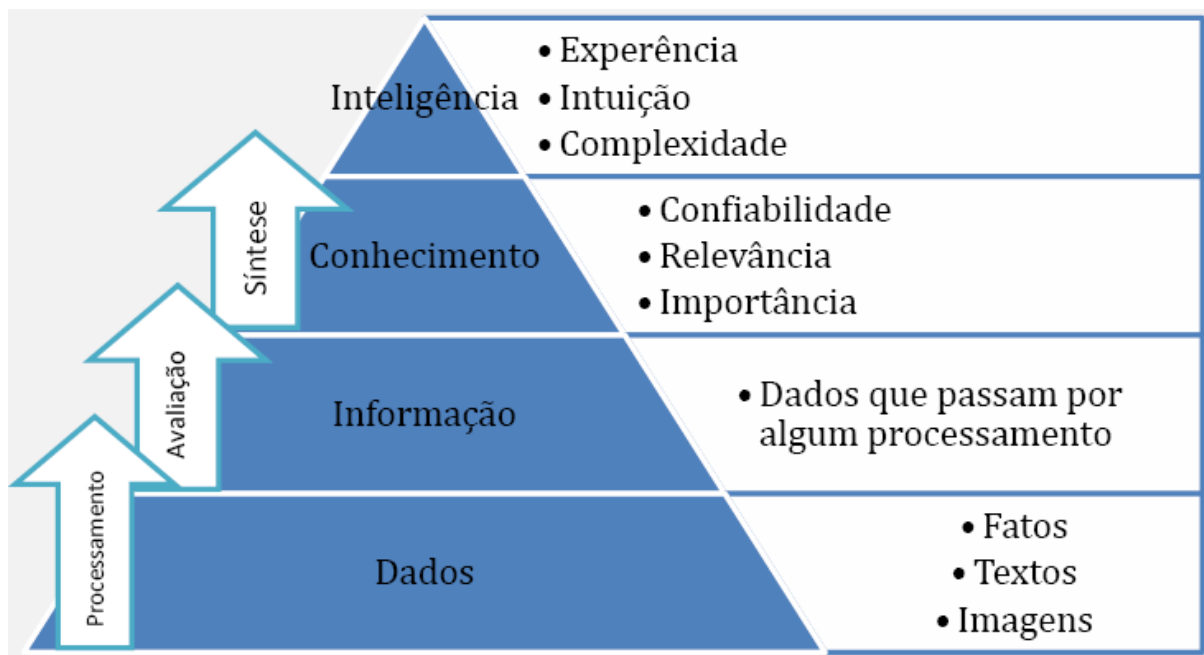
O nível mais alto desta hierarquia é a inteligência, que pode ser entendida como sendo a informação como oportunidade, ou seja, o conhecimento contextualmente relevante que permite atuar com vantagem no ambiente considerado. Veja que estamos colocando em prática nosso conhecimento e aplicando-o da melhor forma possível: isso é ser inteligente! Ou seja, o conhecimento foi sintetizado e aplicado a uma determinada situação, ganhando maior profundidade, em relação ao entendimento do contexto, e desempenhando melhor as ações necessárias.

Portanto, a inteligência resulta da síntese de corpos de conhecimentos, são usados julgamento e intuição daquele que toma decisões e uma visualização completa da situação é obtida. Nesta etapa, os tomadores de decisão aplicam ao conhecimento gerado, suas habilidades, suas competências de negócio e vivência na organização, para identificar direções estratégicas, tais como: novos projetos de pesquisa, acordos de cooperação, transferência de tecnologia e ações e reações da concorrência.

Idealmente, o entendimento da situação apoia a tomada de decisão a partir da visualização do cenário e cria as condições para que o planejamento possa ser realizado e as ações efetivadas. Além disso, podem ser revelados fatores críticos, possibilitando a antecipação a eventos, mediante o reconhecimento das consequências de novos ou iminentes efeitos de uma decisão. Por tudo isso, a inteligência deve ser a base do processo decisório, mesmo considerando que raramente é possível alcançar a compreensão total.

A transformação de conhecimento em inteligência é realizada por meio de síntese, sendo uma habilidade puramente humana baseada na experiência e intuição, que vai muito além da capacidade de qualquer sistema especialista ou de inteligência artificial. Síntese simplesmente não pode ser reduzida a procedimentos ou regras, por não considerarem o complexo.

Por fim, a experiência pode ser definida como a efetividade da inteligência de uma organização, que é aperfeiçoada pelas decisões tomadas e considerada geradora de algum tipo de vantagem. Sendo assim, a experiência agrega valor ao processo decisório de uma organização, por refletir toda a capacidade em atuar no ambiente competitivo.



6. BUSINESS INTELLIGENCE

O conjunto de **arquiteturas, ferramentas, bancos de dados, ferramentas de análise, aplicações e metodologias** que dão suporte às decisões gerenciais por meio de **informações internas e externas** às organizações.

- É um processo amplo que envolve a coleta, análise, compartilhamento, monitoramento dos dados e, por fim, a ação.
- Em outras palavras, trata-se de um **conjunto de técnicas** que visa transformar um grande volume de dados existente na organização em informação útil para o gestor.
- Auxiliando na tomada de **decisão estratégica** e na **descoberta de novas oportunidades**.
- Com as técnicas de BI, é possível ter um **histórico das operações, uma visão do presente e também possíveis previsões para o futuro**.
- Essas técnicas são capazes de lidar com uma **grande quantidade de dados estruturados e não estruturados para criar novas estratégias de negócio baseadas em oportunidades**. Usualmente, as aplicações de BI utilizam os dados que são armazenados nos data warehouses que, por sua vez, utilizam a modelagem multidimensional.

Essas tecnologias têm um profundo impacto na estratégia corporativa, na performance e na competitividade.

Seus principais objetivos incluem:

1. Permitir o acesso interativo, por vezes em tempo real, aos dados.
2. Permitir a manipulação de dados.
3. Dar aos gestores e analistas a capacidade de realizar análise adequada.

O BI tem quatro grandes componentes:

1. **Um data warehouse (DW)** com seus dados-fonte utilizados para a análise de negócios.
2. **A análise de negócio ou business analytics**, uma coleção de ferramentas para manipular e analisar os dados no data warehouse, incluindo data mining.
3. **Business performance management (BPM)** para monitorar e analisar

indicadores de desempenho

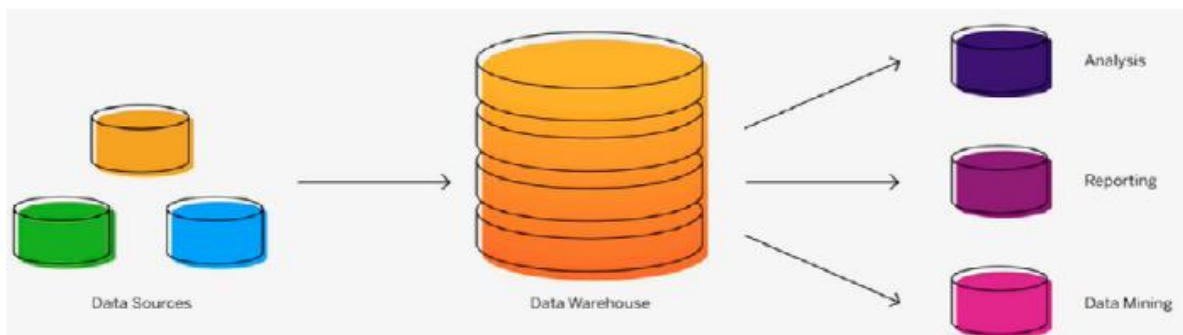
4. Uma interface de usuário fornece uma capacidade visual para os dados solicitados pelos tomadores de decisão. (como o dashboard, cockpit ou portal).

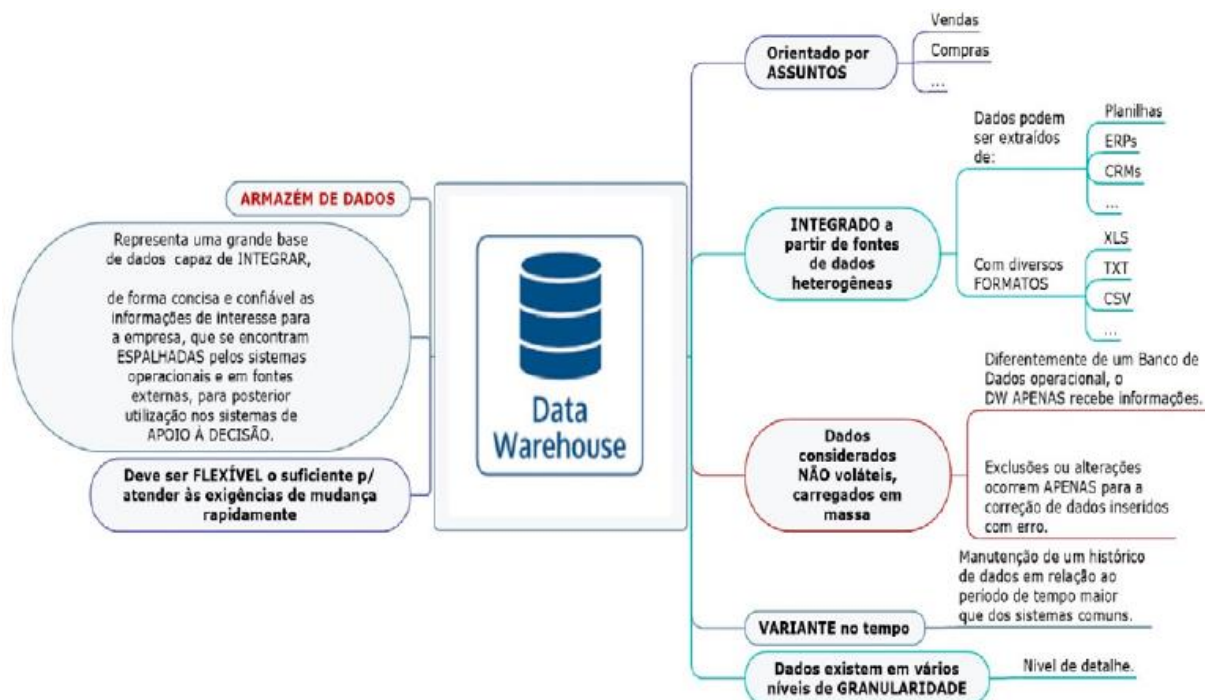
- BI's requerem atenção especial para que após a integração, haja uniformidade dos domínios de todas as colunas do BI que tenham a mesma natureza (por exemplo, cor, idade, etc).
 - A uniformidade dos domínios é essencial para garantir a consistência e a confiabilidade dos dados no processo de Business Intelligence. Quando várias fontes de dados são combinadas, é importante que os dados sejam padronizados e que os domínios de colunas semelhantes sejam consistentes. Isso garante que a análise e a tomada de decisões baseadas nesses dados sejam precisas e significativas.

7. DATA WAREHOUSE

Um *Data Warehouse* (DW) é um sistema de armazenamento que conecta e harmoniza grandes quantidades de dados de muitas formas diferentes. O Data Warehouse pode ser considerado como uma coleção de dados orientada por assunto, integrada, não volátil, variante no tempo, que dá apoio às decisões da administração.

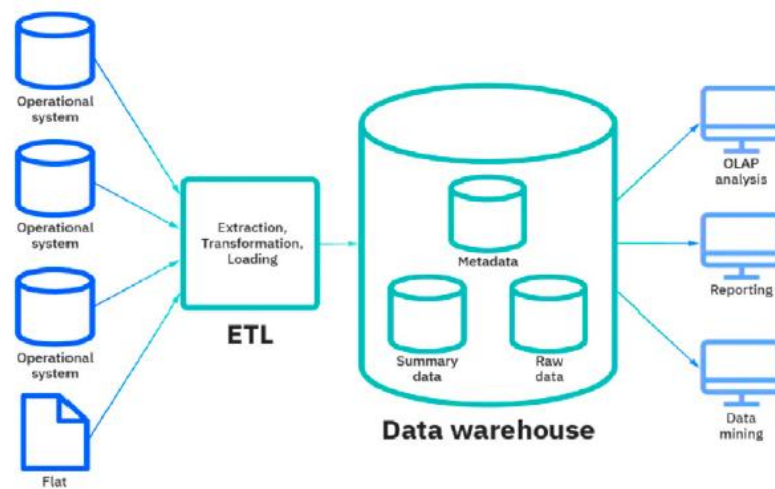
- possibilitar a sua construção a partir de fontes de dados tanto internas quanto externas à organização.





Objetivo DW: Tem como objetivo alimentar a inteligência de negócios (Business Intelligence), relatórios e análises e oferecer suporte aos requisitos de negócio, para que as empresas possam transformar seus dados em insights e tomar decisões inteligentes baseadas em dados.

Os DWs armazenam dados atuais e históricos em um único lugar.



- Durante o processo de ETL, é utilizado um repositório temporário de dados chamado **Data Staging Area**. Ao final do ETL, os dados são carregados no Data Warehouse, onde ficarão armazenados de forma persistente.
 - após serem extraídos das diversas fontes, os dados serão carregados temporariamente no *data staging area* para que possam passar pela etapa de transformação. Ao final do ETL, os dados irão ser carregados no Data Warehouse, onde ficarão armazenados de forma persistente.

DWs modernos são projetados para lidar com dados estruturados e não estruturados, como vídeos, arquivos de imagens e dados de sensor (embora os *Data Lakes* ainda sejam opções melhores para dados não estruturados).

Sem DW é muito difícil combinar dados de fontes heterogêneas, garantir que estejam no formato certo para análise e obter uma visão atual e de longo alcance dos dados ao longo do tempo.



7.1. Características de Datawarehouse



ORIENTADO POR ASSUNTO:

Data Warehouse armazena informações sobre temas específicos importantes para o negócio da empresa. São exemplos típicos de temas: produtos, atividades, contas, clientes. Em contrapartida, quando observamos o ambiente operacional percebemos que ele é organizado por aplicações funcionais. Por exemplo, em uma organização bancária, estas aplicações incluem empréstimos, investimentos e seguros.

INTEGRADO:

refere-se à consistência de nomes, das unidades, das variáveis etc. É importante que os dados armazenados sejam transformados até um estado uniforme. Por exemplo, considere sexo como um elemento de dado. Uma aplicação pode codificar sexo como M/F, outra como 1/0 e uma terceira como H/M. Conforme os dados são inseridos ou repassados para o Data Warehouse, eles são convertidos para um mesmo padrão. O atributo Sexo, portanto, seria codificado apenas de uma forma.

- Da mesma maneira, se um elemento de dado é medido em centímetros em uma aplicação, em polegadas em outra, ele será convertido para uma representação única ao ser colocado no Data Warehouse. Vejam na figura abaixo a ideia por trás do conceito de integração.

NÃO VOLÁTIL:

significa que o Data Warehouse permite apenas a carga inicial dos dados e consultas a estes dados. Após serem integrados e transformados, os dados são carregados em bloco para o DW, para que estejam disponíveis aos usuários para acesso.

- No ambiente operacional, ao contrário, os dados são, em geral, atualizados registro a registro, em múltiplas transações. Esta volatilidade requer um trabalho

considerável para assegurar integridade e consistência através de atividades de rollback, recuperação de falhas, commits e bloqueios. Vejam abaixo uma figura que representa os diferentes ambientes e suas respectivas operações sobre os dados.

VARIANTE NO TEMPO:

trata do fato de um registro em um Data Warehouse referir-se a algum momento específico, significando que ele não é atualizável. Enquanto o dado de produção é atualizado de acordo com mudanças de estado do objeto em questão, refletindo, em geral, o estado do objeto no momento do acesso, em um DW, a cada ocorrência de uma mudança, uma nova entrada é criada para marcar esta mudança.

- O tratamento de séries temporais apresenta características específicas, que adicionam complexidade ao ambiente do Data Warehouse. Deve-se considerar não apenas que os dados tenham uma característica temporal, mas também os metadados, que incluem definições dos itens de dados, rotinas de validação, algoritmos de derivação, etc. Sem a manutenção do histórico dos metadados, as mudanças das regras de negócio que afetam os dados no DW são perdidas, invalidando dados históricos.

GRANULARIDADE DE DADOS:

refere-se ao nível de sumarização dos elementos e de detalhe disponíveis nos dados, considerado o mais importante aspecto do projeto de um Data Warehouse. Em um nível de granularidade muito alto, o espaço em disco e o número de índices necessários se tornam bem menores, há, porém, uma diminuição da possibilidade de utilização dos dados para atender a consultas detalhadas.

CREDIBILIDADE DOS DADOS:

É necessário, para que as análises sejam consideradas corretas que os dados tenham baixa dispersão e estejam consistentes com os valores observados. É importante lembrar que serão feitas várias manipulações nos dados, inclusive agregações e cálculos estatísticos. A depender dos desvios presentes nos dados as análises podem ser inócuas ou incoerentes com a realidade.



7.2. Formas de Armazenamento OLAP

Em um Data Warehouse (Armazém de Dados) existem algumas formas distintas para armazenar dados OLAP (Online Analytical Processing), conhecidas pelos seguintes acrônimos:

As formas de armazenamento são:

- **Relacional (ROLAP)** - Usa banco de dados relacional para realizar as análises. Manipula fatos atômicos.
- **Multidimensional (MOLAP)** - Armazena de maneira multidimensional para visualizar de maneira multidimensional. Manipula fatos agregados.
- **Híbrido (HOLAP)** - Combina os tipos ROLAP e MOLAP. Manipula fatos atômicos e agregados.
- **Desktop (DOLAP)** - Armazena dados multidimensionais no servidor. A análise de informações é realizada no cliente, após consulta ao servidor.

8. MODELO MULTIDIMENSIONAL

MODELO DIMENSIONAL:

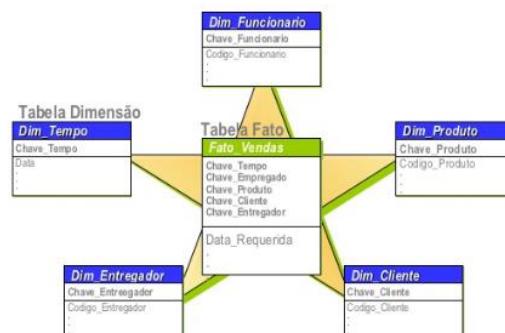
- contém as mesmas informações que um modelo normalizado.
- organizar os dados com um propósito diferente
- possuem as seguintes preocupações: facilidade de compreensão ao usuário, desempenho da consulta e resiliência às mudanças.



Figura 1 - Preocupações do modelo dimensional

- A modelagem multidimensional, ou dimensional como às vezes é chamada, é a técnica de modelagem de banco de dados para o auxílio às consultas em um Data Warehouse nas mais diferentes perspectivas. 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: Fatos e Dimensões. Ambos são obrigatórios e possuem características complementares dentro de um Data Warehouse. As Dimensões são os descritores dos dados oriundos dos Fatos. Possui o caráter qualitativo da informação. É a Dimensão que permite a visualização das informações por diversos aspectos e perspectivas.

- Os Fatos servem para o armazenamento dos registros e medidas (quase sempre) numéricas associadas a eventos de negócio. Perceba que até aqui não falei em tabela, mas por quê? Fatos e dimensões são elementos genéricos que existem nos modelos dimensionais, mas estamos acostumados a falar da implementação ou estruturação relacional deles. Neste contexto, aparecem os conceitos tabela fato e tabelas dimensões
- As Dimensões são os descritores dos dados oriundos da Fato. Possui o caráter qualitativo da informação e relacionamento de “um para muitos” com a tabela Fato. Ou seja, cada linha da tabela dimensão ligada diretamente a tabela fato pode estar associada a várias linhas da tabela dimensão. É a Dimensão que permite a visualização das informações por diversos aspectos e perspectivas.
- NÃO existe remoção e atualização em Data Warehouse onde fica a modelagem dimensional (a não ser por erro e quando o dado chega no seu fim de ciclo de vida)
- As Fatos contêm as métricas. Possui o caráter quantitativo das informações descritivas armazenadas nas Dimensões. É onde estão armazenadas as ocorrências do negócio e possui relacionamento de “muitos para um” com as tabelas periféricas (Dimensão). Uma tabela fato armazena as medições de desempenho decorrentes de eventos dos processos de negócios de uma organização. Basicamente, representa uma medida de negócios. Uma tabela fato contém vários fatos, correspondentes a cada uma das suas linhas. Cada linha corresponde a um evento de medição. Os dados em cada linha estão a um nível específico de detalhe, referido como o grão, por exemplo, uma linha por produto vendido em determinada loja em um dia específico do não.



- FATO: numérico/valor/chaves
- DIMENSÃO: qualitativo/descritivo

- Uma única linha da tabela fato tem uma relação um-para-um com o evento de medição, como descrito pela granularidade da tabela fato. Veja que estamos falando de um evento de medição, ou seja, algo que aconteceu e precisa ser armazenado para análise posterior.
- Uma tabela fato corresponde a um evento físico observável, e não às exigências de um relatório específico. Dentro de uma tabela fatos, apenas fatos consistentes com a granularidade definida são permitidos. Por exemplo, em uma transação de vendas no varejo, a quantidade de um produto vendido e seu preço são bons fatos. Veja a figura abaixo que representa uma tabela fato de vendas:

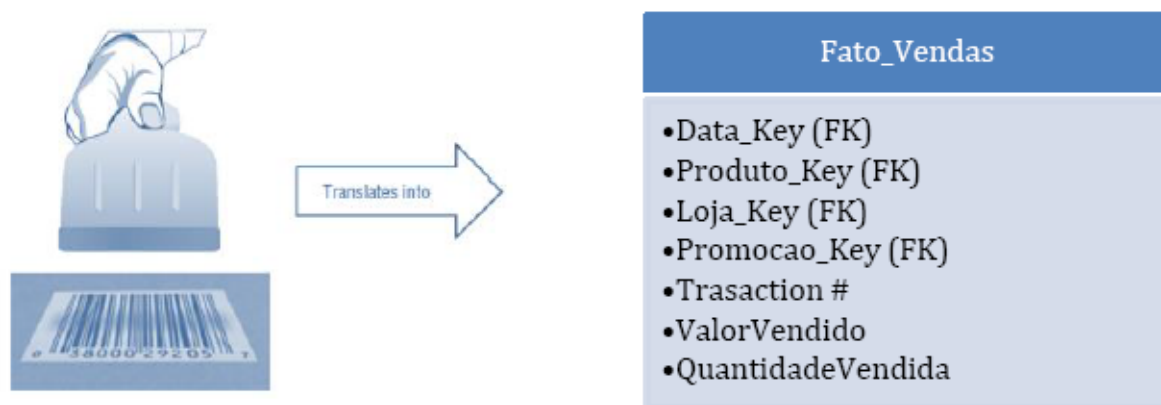


Figura 2 - Tabela fato de vendas

A ideia de que um evento de medição no mundo físico tenha uma relação de um-para-um com uma única linha na tabela fato é um princípio fundamental para a modelagem dimensional. Todo o resto constrói a partir desta fundação.

Aditivos	Semi-aditivos	Não aditivos
<ul style="list-style-type: none"> • Podem ser agrupadas em uma qualquer das dimensões associadas à tabela de fatos • Os fatos mais flexíveis e úteis. • Lucro líquido 	<ul style="list-style-type: none"> • Podem ser agrupadas em algumas dimensões, mas não todas. • Ex.: Saldo em conta - não podem ser resumidos por meio da dimensão de tempo 	<ul style="list-style-type: none"> • Nunca podem ser adicionados ou somados • Taxas e percentuais

Figura 3 - Tipos de medidas presentes nas tabelas fatos

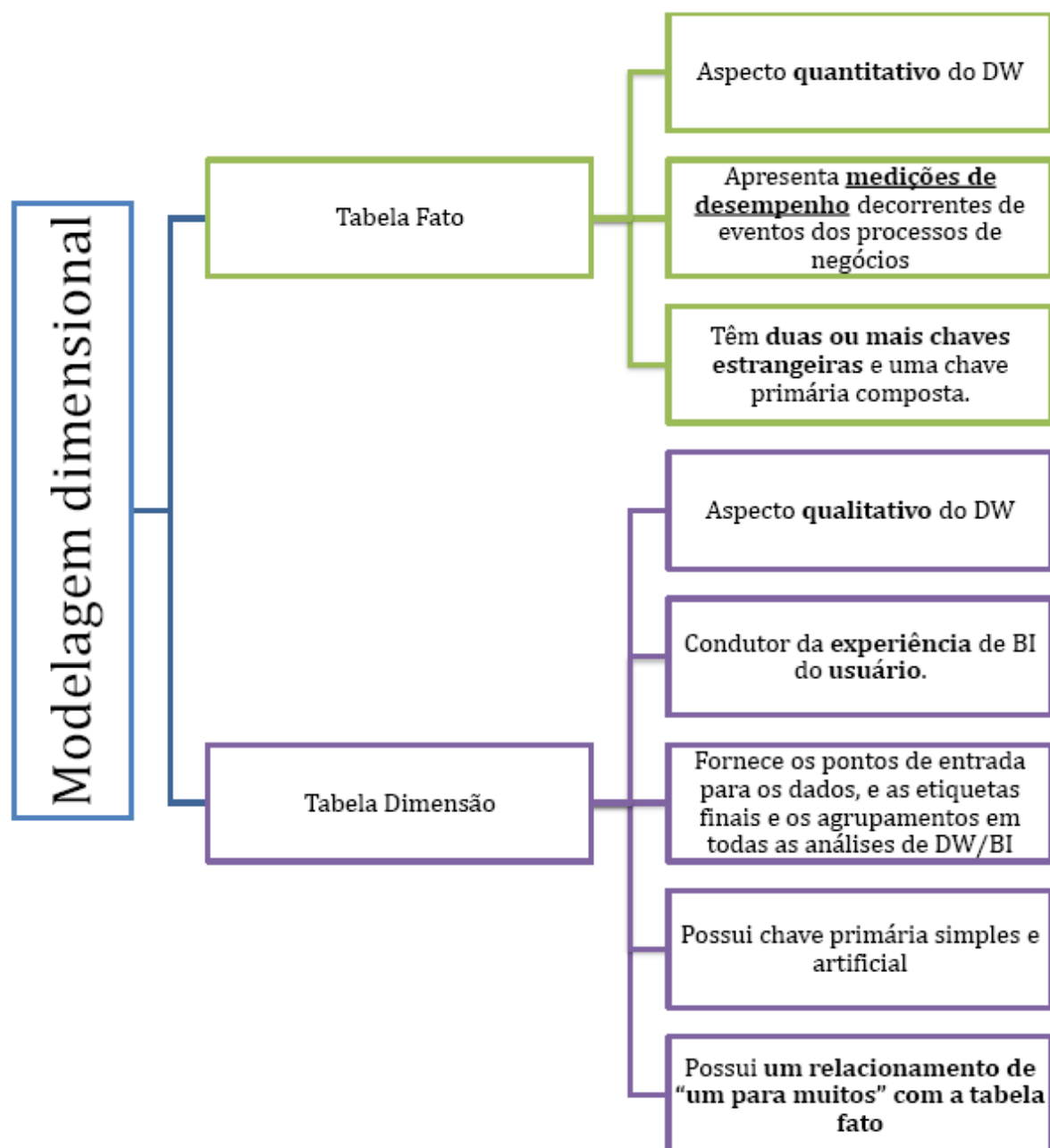



Figura 5 - Esquema das tabelas fato e dimensões

10.1. TIPOS DE FATOS NUMÉRICOS

- **Fatos aditivos:** são os fatos mais flexíveis, pois podem ser somados em todas as dimensões. Por exemplo: número de vendas. Podemos somar o número de vendas por produto, sumarizar por loja ou ainda por período de tempo;
- **Fatos semi-aditivos:** podem ser somados em apenas algumas dimensões, mas não em todas. Por exemplo: o balanço diário do caixa. Faz sentido somar esse dado de todas as lojas de um estado para saber qual o balanço diário sumarizado daquela região. No entanto, não faz sentido somar os balanços de todos os dias dentro de um mês. Percebe que isso não traz nenhum significado útil?
- **Fatos não-aditivos:** não podem ser somados em nenhuma dimensão. Por exemplo: margem de lucro. Um produto A tem margem de lucro de 70%, enquanto o produto B tem margem de lucro de 40%. Se somarmos esses valores chegamos a 110%, o que não tem significado nenhum.

10.2. ESQUEMAS MULTIDIMENSIONAIS

10.2.1 Modelo Estrela

	Modelo estrela
	Uma tabela fato central e dimensões respresentadas em apenas 1 tabela.
	Todas as dimensões são ligadas diretamente a tabela fato.
	As tabelas dimensões são desnormalizadas .
	Mais fácil de ser entendido pelos usuários finais. (consultas mais simples)
Performance melhor do que o modelo snowflake	

10.2.2 Modelo Floco de Neve

Floco de neve	Dimensões normalizaadas (pelo menos uma) até a 3FN.
	Apresenta hierarquia nas dimensões.
	Algumas dimensões não estão ligadas diretamente a tabela fato.
	Modelo mais complexo, dificulta o entendimento por usuários finais.
	Ocupa menos espaço de armazenamento.
	Consultas mais complexas e lentas quando comparadas com o modelo estrela.

10.2.3 Resumo dos Esquemas Multidimensionais

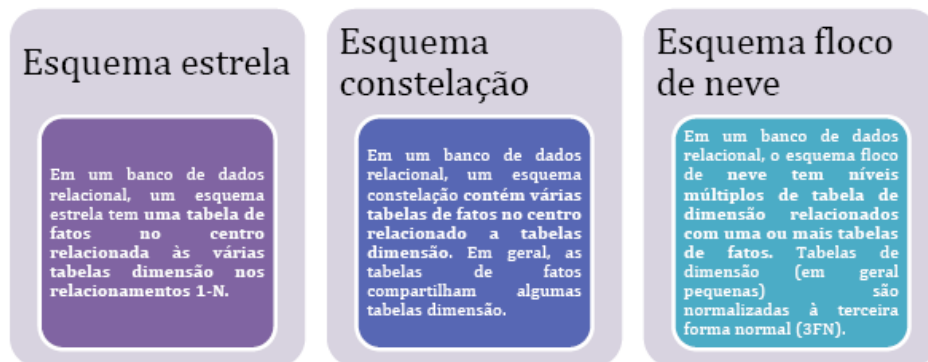
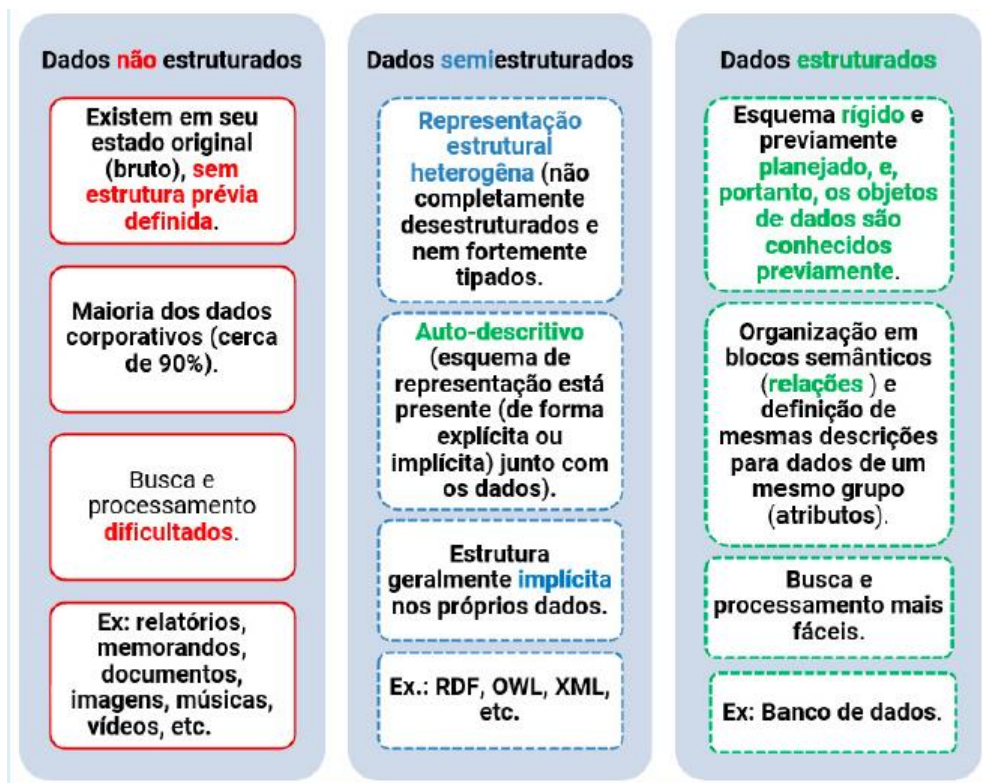


Figura 11 - Resumo dos esquemas multidimensionais

11. DADOS ESTRUTURADOS X DADOS NÃO ESTRUTURADOS

TIPOS DE DADOS	DESCRIÇÃO
DADOS ESTRUTURADOS	São dados que podem ser armazenados, acessados e processados em formato fixo e padronizado de acordo com alguma regra específica. Esta organização é geralmente feita por colunas e linhas (semelhante a planilhas do Excel), mas pode variar de acordo com a fonte de dados. Exemplo: Planilhas Eletrônicas, Bancos de Dados Relacionais e CSV.
DADOS SEMI-ESTRUTURADOS	São dados estruturados que não estão de acordo com a estrutura formal dos modelos de dados como em tabelas, mas que possuem marcadores para separar elementos semânticos e impor hierarquias de registros e campos dentro dos dados Exemplo: Dados de E-mail, Arquivos XML, Arquivos JSON e Banco de Dados NoSQL.
DADOS NÃO-ESTRUTURADOS	São dados que apresentam formato ou estrutura desconhecidos, em que não se sabe extrair de forma simples os valores desses dados em forma bruta. Exemplo: Documentos, Imagens, Vídeos, Arquivos de Texto, Posts em Redes Sociais.





	DATA WAREHOUSE	DATA LAKE
normalizados	Dados geralmente são tratados (limpos, combinados, organizados, etc) antes de serem armazenados.	Dados geralmente são armazenados da maneira que foram capturados – brutos, sem nenhum tratamento.
	Podem armazenar todos os tipos de dados, mas o foco é nos dados estruturados.	Armazenam dados estruturados, semi-estruturados e não-estruturados.
	Ideal para usuários operacionais visto que as ferramentas analíticas são mais fáceis de usar.	Ideal para cientistas de dados visto que as ferramentas analíticas são mais difíceis de usar.
	Armazenamento de dados custam geralmente mais caro e consome mais tempo.	Armazenamento de dados custam geralmente mais barato e consome menos tempo.
	Um esquema é definido antes dos dados serem armazenados.	Um esquema é definido após os dados serem armazenados.
	Armazenam um grande volume de dados.	Armazenam um gigantesco volume de dados.

12. OLAP (On-Line Analytical Processing):

é uma metodologia que visa analisar os dados organizacionais e corporativos de maneira mais ágil, consistente e interativa pelos gestores e quaisquer outros interessados nos dados.

O método examina um grande volume de informações com alto desempenho, de forma dinâmica e através de múltiplas perspectivas (por vezes o método é chamado de análise multidimensional).

O processo OLAP é definido como “a sistemática de criar, gerenciar, analisar e gerar relatórios sobre informações organizacionais”.

Sua característica multidimensional ocorre devido a percepção e manipulação dos dados como se fossem armazenados em um “array multidimensional”.

- O modelo relacional põe os dados em tabelas, enquanto o OLAP usa a representação de arrays multidimensionais.
- conjunto de ferramentas de software que permite aos gerentes e diretores de empresas a terem acesso dinâmico a informações armazenadas nos diversos sistemas corporativos, podendo realizar cruzamentos e análises de informações em tempo real sob diversas perspectivas com o intuito de auxiliar na tomada de decisões.
- Uma ferramenta OLAP é caracterizada pela análise multidimensional dinâmica dos dados, apoiando o usuário final nas suas atividades e permitindo consultas ad-hoc (created or done for a particular purpose as necessary).
- Ela permite que você realize diversas operações em modelos de dados multidimensionais – navegando por dimensões e hierarquias – com o intuito de enxergar informações preciosas que, de alguma forma, te auxiliem na tomada de decisões estratégicas.
- A OLAP abstrai as complexidades de forma que qualquer pessoa consiga manipular essa ferramenta.

10.3. Níveis de análise de dados: DESCRITIVO, PREDITIVO e PRESCRITIVO.

1. **ANÁLISE DE DADOS DESCRITIVA (ou de EXTRAÇÃO DE RELATÓRIOS):** diz respeito a conhecer o que está acontecendo na organização e entender tendências e causas subjacentes de tais ocorrências. Em primeiro lugar, isso envolve a consolidação de fontes de dados e a disponibilidade de todos os dados relevantes de um modo que permita a extração e a análise apropriadas de relatórios. Geralmente, o desenvolvimento dessa infraestrutura de dados faz parte dos *data warehouses*. A partir dessa infraestrutura de dados, podemos desenvolver relatórios, consultas, alertas e tendências apropriados usando ferramentas e técnicas de extração de relatórios.
2. **ANÁLISE DE DADOS PREDITIVA:** visa determinar o que é mais provável de acontecer no futuro. Essa análise se baseia em técnicas estatísticas, bem como em outras técnicas desenvolvidas mais recentemente que recaem na categoria geral de mineração de dados. A meta dessas técnicas é conseguir prever se o cliente está propenso a migrar para um concorrente, o que o cliente tende a comprar a seguir e em qual quantidade, a quais promoções o cliente reagiria, qual o risco de crédito do cliente, e assim por diante.
3. **ANÁLISE DE DADOS PRESCRITIVA:** a meta dessa categoria de análise de dados é reconhecer o que está acontecendo ("*...obter informações sobre hábitos de compra...*"), bem como o que deve vir a acontecer, e tomar decisões para garantir o melhor desempenho possível. A meta aqui é chegar a uma decisão ou a uma recomendação ("*...recomendar os melhores cupons de desconto...*") para uma ação específica. Tais recomendações podem assumir a forma de uma decisão pontual do tipo sim/não a um problema, uma quantidade específica (o preço de um determinado item, digamos) ou um conjunto completo de planos de produção. A decisão pode ser apresentada a um tomador de decisões ou pode ser usada diretamente em um sistema automatizado de regras decisórias (como em sistemas de precificação de companhias aéreas). Por isso, esses tipos de análise de dados também podem ser denominados análise de dados normativa ou decisória.

10.4. Operações OLAP

Existem muitas operações (métodos) com o objetivo de manipular e permitir alterar a perspectiva de observação dos dados, visando dar uma visualização diferente aos usuários. Entre as operações existem:

- **Pivot** – Também chamado de *rotation*, executa uma operação de visualização rotacional dos eixos de um determinado cubo, mudando o eixo de visualização. Ou seja, através da operação *pivot* é possível alterar o eixo de observação dos dados, permitindo visualizar as informações através de uma nova perspectiva
 - **rotação da perspectiva, mudando de produtos X estados para estados X produtos.**
- **Slice** – Através do comando *slice* é possível selecionar os dados de uma das dimensões de um cubo. Ou seja, é possível selecionar as informações de uma dimensão do cubo e caso necessário extrair através de um método de seleção.
- **Roll up - roll up ou drill Up** é o método que realiza a operação contrária ao *drill Down*, aumentando a granularidade e com isso reduzindo o nível detalhe dos dados, ou seja, as informações são resumidas de forma generalizada crescente.
- **Drill Down** – é o método que realiza a operação contrária ao *roll Up* ou *Drill Up*, ou seja, enquanto a operação *roll Up* aumenta a granularidade das informações, reduzindo com isso o nível de detalhes das informações, o *Drill Down* reduz a granularidade, aumentando o nível de detalhe dos dados, ou seja, a generalização das informações é reduzida.

OLTP (ONLINE TRANSACTION PROCESSING)	OLAP (ONLINE ANALYTICAL PROCESSING)
<ul style="list-style-type: none">▪ Sistema de gerenciamento de transações em um banco de dados.▪ Foco no nível operacional da organização, visando a execução rotineira do negócio.▪ Tabelas formadas por linhas e colunas e geralmente normalizadas.▪ Lidam com Bancos de Dados Transacionais em geral estruturados em um modelo relacional.▪ Executados de forma mais rápida com tempo de resposta de milissegundos até segundos.▪ Apresentam dados detalhados (baixa granularidade).	<ul style="list-style-type: none">▪ Sistema de gerenciamento de consultas e análise de dados.▪ Foco no nível estratégico da organização, visando a análise empresarial e a tomada de decisão.▪ Tabelas formadas por fatos, dimensões e medidas e geralmente desnormalizadas.▪ Lidam com Bancos de Dados Dimensionais (DW/DM) em geral estruturados em modelo dimensional.▪ Executados de forma mais lenta com tempo de resposta de segundos até horas.▪ Apresentam dados sumarizados (alta granularidade).
<ul style="list-style-type: none">▪ Atualizações de dados são realizadas no momento de cada transação e são altamente frequentes.▪ Não é otimizado para lidar com uma grande quantidade de dados (baixo armazenamento)▪ Dados voláteis e passíveis de inserção, alteração ou exclusão.▪ São orientados a registros ou tuplas e possuem consultas pré-definidas.	<ul style="list-style-type: none">▪ Atualizações de dados são realizadas no processo de carga de dados e são bem menos frequentes.▪ É otimizado para lidar com uma massiva quantidade de dados (alto armazenamento)▪ Dados históricos e não-voláteis, não podendo ser alterados ou excluídos (salvo casos específicos)▪ São orientados a arrays ou vetores e possuem consultas ad-hoc.

Características	OLTP	OLAP
Operação típica	Atualização	Análise
Telas	Imutáveis	Definida pelo usuário
Nível de dados	Atomizado	Sumarizados
Recuperação	Poucos Registros	Muitos registros
Orientação	Registros	Arrays de dados
Modelagem, organização dos dados.	Processos de negócio, por aplicação, sistema de informação	Assuntos/Negócios
Natureza dos dados /Conteúdo	Permitem atualizações (dinâmica), Valores Correntes	Dados históricos, sumarizados e integrados. (estática – não volátil)
Formato das estruturas	Relacional, próprio para computação transacional.	Dimensional, simplificado, próprio para atividades analíticas
Uso	Processamento repetitivo, altamente estruturado em tabelas.	Estruturados em fatos e dimensões, com processamento analítico
Tempo de resposta	Otimizado para faixas abaixo de 1 seg.	Análises mais complexas, com tempos de respostas maiores.

13. BIG DATA

Big Data é uma coleção de conjuntos de dados, grandes e complexos, que não podem ser processados por bancos de dados ou aplicações de processamento tradicionais.

Big Data é definido genericamente como a captura, gerenciamento e análise de dados que vão além dos dados tipicamente estruturados, que podem ser consultados e pesquisados através de bancos de dados relacionais”.

Frequentemente são dados obtidos de arquivos não estruturados como vídeo digital, imagens, dados de sensores, arquivos de logs e de qualquer tipo de dados não contidos em registros típicos com campos que podem ser pesquisados.

Big Data tem dados estruturados e não estruturados! Big Data é o termo que descreve o imenso volume de dados – estruturados e não estruturados – que impactam os negócios no dia a dia.

Análise de Big Data é...

Uma estratégia baseada em tecnologia que **permite a coleta de insights mais profundos e relevantes dos clientes, parceiros e sobre o negócio** — ganhando assim uma vantagem competitiva.

Trabalhar com conjuntos de dados cujo o porte e variedade estão **além da habilidade de captura, armazenamento e análise de softwares de banco de dados típicos**.

Processamento de um **fluxo contínuo de dados em tempo real**, possibilitando tomada de decisões sensíveis ao tempo mais rápido do que em qualquer outra época.

Distribuído na natureza. O processamento de análise vai aonde estão os dados para maior velocidade e eficiência.

Um novo paradigma no qual a Tecnologia da Informação (TI) colabora com usuários empresariais e "cientistas de dados" para identificar e implementar análises que ampliam a eficiência operacional e resolvem novos problemas empresariais.

Transferir a tomada de decisão dentro da empresa e permitir com que as pessoas tomem decisões melhores, mais rápidas e em tempo real.

Análise de Big Data **NÃO** é...

Só tecnologia. No nível empresarial, refere-se a explorar as amplamente melhoradas fontes de dados para ganhar *insights*.

Somente volume. Também se refere à variedade e velocidade. Mas, talvez mais importante, refere-se ao valor derivado dos dados.

Mais gerada ou utilizada somente por grandes empresas online como Google ou Amazon. Embora as empresas de internet possam ter sido pioneiras no Big Data na escala web, **aplicativos chegam a todas as indústrias**.

Uso de bancos de dados relacionais tradicionais "tamanho único" criados com base em disco compartilhado e arquitetura de memória. Análise de Big Data usa uma rede de recursos de computação para processamento maciçamente paralelo (PMP).

Um substituto de bancos de dados relacionais ou centros de processamento de dados. Dados estruturados continuam a ser de importância crítica para as empresas. No entanto, sistemas tradicionais podem não ter capacidade de manipular as novas fontes e contextos do Big Data.

13.1. Objetivo Big Data

"o objetivo principal do Big Data é obter informação útil a partir de dados armazenados em "tempo real" (espontâneos) e por isso esses dados não são estruturados, o que torna a aplicação de técnicas de extração de informação mais difícil! Assim, estamos falando de muitos dados que são gerados e consumidos rapidamente.

É por isso que dizemos que as características mais marcantes do Big Data são:

- volume, e
- velocidade.



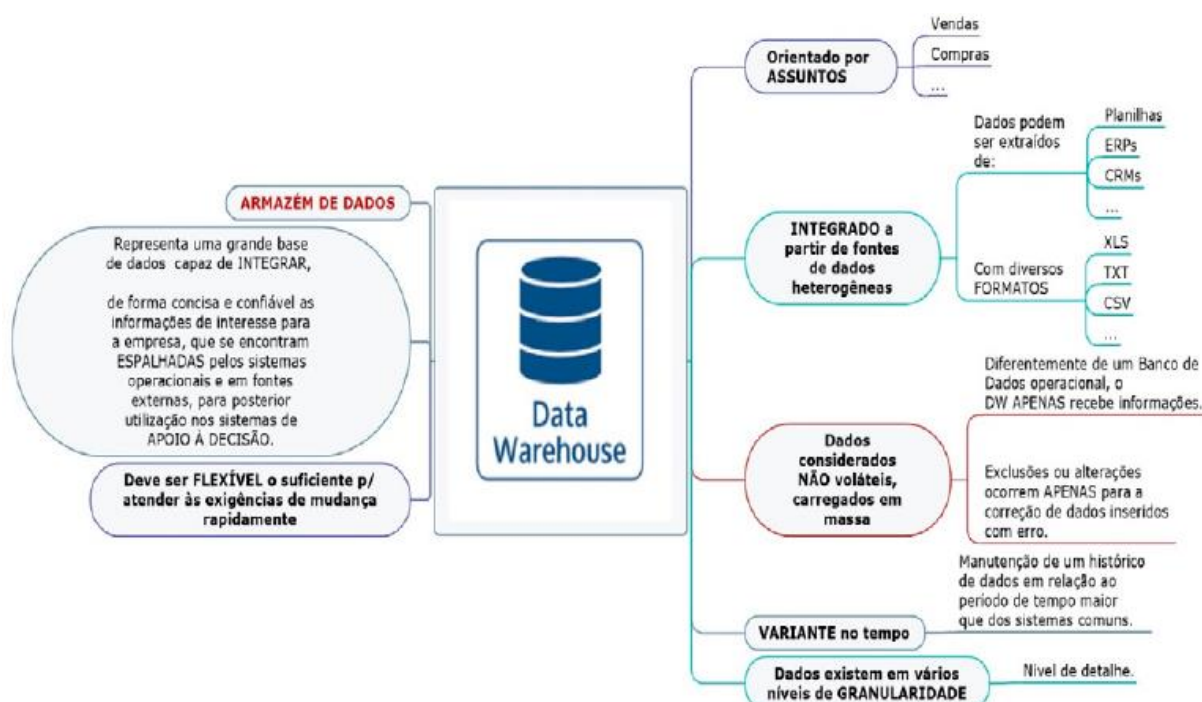
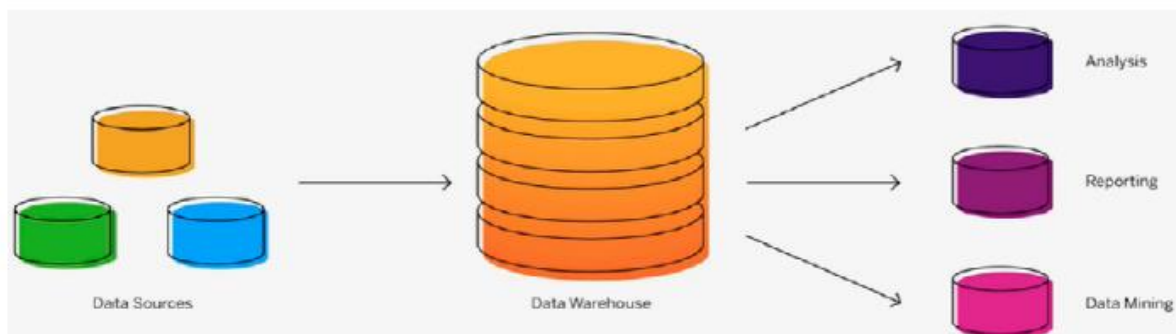
13.2. Armazenamento BIG DATA

Os dados são estruturados ou podem ser estruturados antes do armazenamento?
Usamos um **Data Warehouse**!

Os dados **não** são estruturados ou **não** podem ser estruturados antes do armazenamento?
Usamos um **Data Lake** ou um **Data Store**!

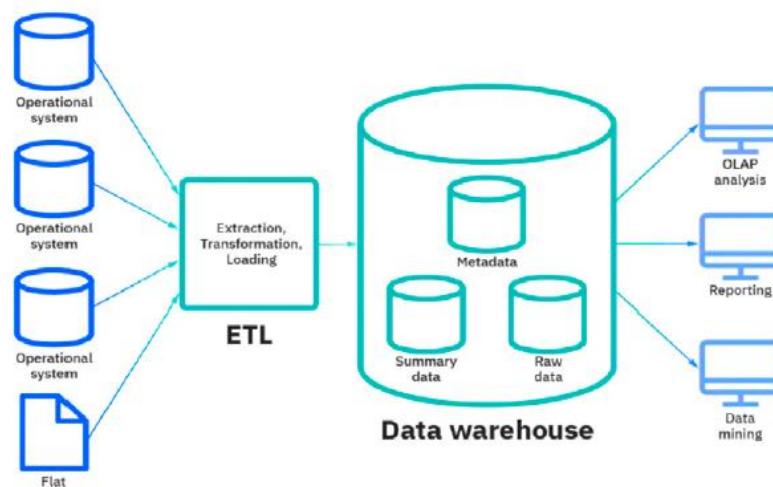
13.2.1 Data Warehouse

Um *Data Warehouse* (DW) é um sistema de armazenamento que conecta e harmoniza grandes quantidades de dados de muitas formas diferentes. O Data Warehouse pode ser considerado como uma coleção de dados orientada por assunto, integrada, não volátil, variante no tempo, que dá apoio às decisões da administração.



Objetivo DW: Tem como objetivo alimentar a inteligência de negócios (Business Intelligence), relatórios e análises e oferecer suporte aos requisitos de negócio, para que as empresas possam transformar seus dados em insights e tomar decisões inteligentes baseadas em dados.

Os DWs armazenam dados atuais e históricos em um único lugar.



DWs modernos são projetados para lidar com dados estruturados e não estruturados, como vídeos, arquivos de imagens e dados de sensor (embora os *Data Lakes* ainda sejam opções melhores para dados não estruturados).

Sem DW é muito difícil combinar dados de fontes heterogêneas, garantir que estejam no formato certo para análise e obter uma visão atual e de longo alcance dos dados ao longo do tempo.

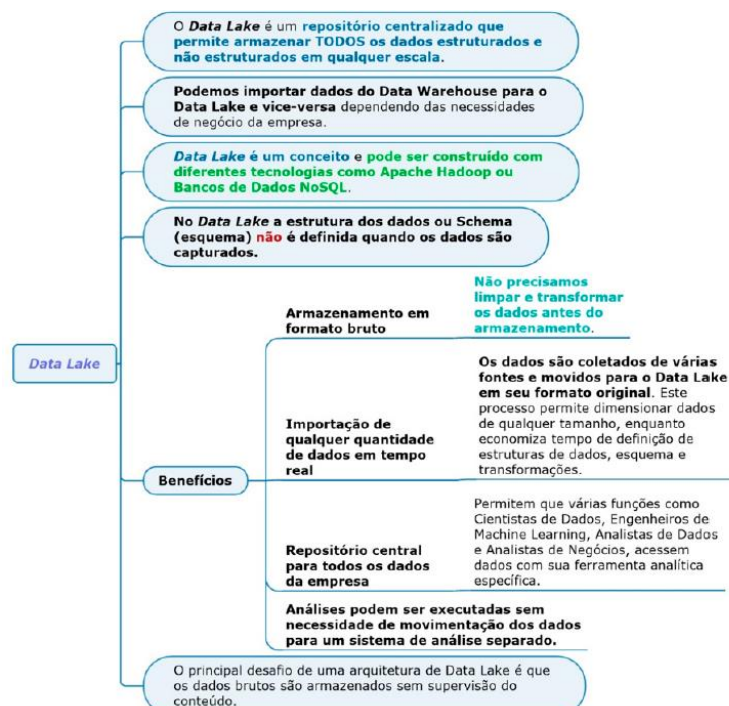


13.2.2 Data Lake

O *Data Lake* é um repositório centralizado que permite armazenar TODOS os dados estruturados e não estruturados em qualquer escala. Pode-se armazenar os dados como estão na fonte, sem ter que primeiro estruturá-los e executar diferentes tipos de análises – de painéis e visualizações a processamento de Big Data, análises em tempo real e aprendizado de máquina para orientar melhores decisões

Dependendo dos requisitos, uma empresa típica exigirá um *Data Warehouse* e um *Data Lake*, pois eles atendem a diferentes necessidades e casos de uso.

A estrutura dos dados ou *Schema* (esquema) não é definida quando os dados são capturados. Dessa forma, pode-se armazenar todos os dados em formato bruto sem a necessidade de saber quais perguntas de negócio deverão ser respondidas no futuro



13.3. TIPOS DE ANÁLISE BIG DATA:



BIG DATA ANALYTICS É:	BIG DATA ANALYTICS NÃO É:
Uma estratégia baseada em tecnologia que permite coletar insights mais profundos e relevantes de clientes, parceiros e negócio, ganhando assim uma vantagem competitiva.	Somente tecnologia – no nível empresarial, refere-se a explorar fontes amplamente melhoradas de dados para adquirir insights.
Trabalhar com conjuntos de dados cujo porte e variedade estão além da habilidade de captura, armazenamento e análise de softwares de banco de dados típicos.	Somente volume – também se refere à variedade e à velocidade, mas – talvez mais importante – refere-se ao valor derivado dos dados.
Processamento de um fluxo contínuo de dados em tempo real, possibilitando a tomada de decisões sensíveis ao tempo mais rápido do que em qualquer outra época.	Mais gerada ou mais utilizada somente por grandes empresas online como Google ou Amazon. Embora as empresas de internet possam ter sido pioneiras no Big Data na escala web, aplicativos chegam a todas as indústrias.
Distribuído na natureza, isto é, o processamento de análise vai aonde estão os dados para maior velocidade e eficiência.	Uso de bancos de dados relacionais tradicionais de “tamanho único” criados com base em disco compartilhado e arquitetura de memória. Análise de Big Data usa uma rede de recursos de computação para processamento massivamente paralelo e escalável.
Um novo paradigma no qual a tecnologia da informação colabora com usuários empresariais e “cientistas de dados” para identificar e implementar análises que ampliam a eficiência operacional e resolvem novos problemas empresariais.	Um substituto de bancos de dados relacionais – dados estruturados continuam a ser de importância crítica para as empresas. No entanto, sistemas tradicionais podem não ter capacidade de manipular as novas fontes e contextos do Big Data.
Transferir a tomada de decisão dentro da empresa e permitir que pessoas tomem decisões melhores, mais rápidas e em tempo real.	-

Análise de Big Data é...

Uma estratégia baseada em tecnologia que **permite a coleta de insights mais profundos e relevantes dos clientes, parceiros e sobre o negócio** — ganhando assim uma vantagem competitiva.

Trabalhar com conjuntos de dados cujo o porte e variedade estão além da habilidade de captura, armazenamento e análise de softwares de banco de dados típicos.

Processamento de um fluxo contínuo de dados em tempo real, possibilitando tomada de decisões sensíveis ao tempo mais rápido do que em qualquer outra época.

Distribuído na natureza. O processamento de análise vai aonde estão os dados para maior velocidade e eficiência.

Um novo paradigma no qual a Tecnologia da Informação (TI) colabora com usuários empresariais e "cientistas de dados" para identificar e implementar análises que ampliam a eficiência operacional e resolvem novos problemas empresariais.

Transferir a tomada de decisão dentro da empresa e permitir com que as pessoas tomem decisões melhores, mais rápidas e em tempo real.

Análise de Big Data **NÃO** é...

Só tecnologia. No nível empresarial, refere-se a explorar as amplamente melhoradas fontes de dados para ganhar *insights*.

Somente volume. Também se refere à variedade e velocidade. Mas, talvez mais importante, refere-se ao valor derivado dos dados.

Mais gerada ou utilizada somente por grandes empresas online como Google ou Amazon. Embora as empresas de internet possam ter sido pioneiras no Big Data na escala web, **aplicativos chegam a todas as indústrias.**

Uso de bancos de dados relacionais tradicionais "tamanho único" criados com base em disco compartilhado e arquitetura de memória. Análise de Big Data usa uma rede de recursos de computação para processamento maciçamente paralelo (PMP).

Um substituto de bancos de dados relacionais ou centros de processamento de dados. Dados estruturados continuam a ser de importância crítica para as empresas. No entanto, sistemas tradicionais podem não ter capacidade de manipular as novas fontes e contextos do Big Data.

13.4. 5 V's do BIG DATA



13.5.1 VOLUME:

Big Data trata de uma grande quantidade de dados gerada a cada segundo. Pense em todos os e-mails, mensagens de Twitter, fotos e vídeos que circulam na rede a cada instante. Não são terabytes e, sim, zetabytes ou brontobytes. A tecnologia do Big Data serve exatamente para lidar com esse volume massivo de dados, guardando-os em diferentes localidades e juntando-os através de software.

13.5.2 VELOCIDADE:

Refere-se à velocidade com que os dados são criados. São mensagens de redes sociais se viralizando em segundos, transações de cartão de crédito sendo verificadas a cada instante ou os milissegundos necessários para calcular o valor de compra e venda de ações. O Big Data serve para analisar os dados no instante em que são criados, em tempo real, sem ter de armazená-los.

Não apenas o volume de dados é gigantesco, mas a velocidade em que esses dados são produzidos (e se tornam desatualizados é vertiginosa). Justamente por isso o segundo desafio do Big Data é o *timing* do processamento desses dados: para que possuam valor real e aplicabilidade no mercado, é preciso utilizar os dados antes que se tornem desatualizados. O objetivo, portanto, é alcançar formas de trabalhar o processamento dessas informações em tempo real.

13.5.3 VARIEDADE:

No passado, a maior parte dos dados utilizados por organizações era estruturado e podia ser facilmente armazenado em tabelas de bancos de dados relacionais. No entanto, a maioria dos dados do mundo não se comporta dessa forma. Com o Big Data, mensagens, fotos, mídia social, e-mail, vídeos e sons – que são dados não-estruturados – podem ser administrados juntamente com dados tradicionais.

Os dados de que dispomos atualmente são provenientes das mais diversas fontes: redes sociais, aplicativos, cookies, IoT, e-mails, etc. Isso significa que não seguem um único padrão e nem fornecem todos o mesmo tipo de informações, tornando a tarefa de compilar esses dados em um banco de dados tradicional inviável. É preciso desenvolver novas ferramentas de análise que respondam à heterogeneidade dos dados.

13.5.4 VERACIDADE

Um dos pontos mais importantes de qualquer informação é que ela seja verdadeira. Com o Big Data, não é possível controlar cada hashtag do Twitter ou notícia falsa na internet, mas com análises e estatísticas de grandes volumes de dados é possível compensar as informações incorretas. Dentre a massa de dados que circula, é preciso estabelecer quais os dados que são verídicos e que ainda correspondem ao momento atual.

Dados desatualizados podem ser considerados inverídicos, mas não porque tenham sido gerados com segundas intenções, mas porque não correspondem mais à realidade e podem guiar uma empresa a decisões equivocadas. O desafio posto pelo Big Data é, então, determinar a relevância dos dados disponíveis para uma empresa, de forma que essas informações possam servir de guia para o seu planejamento com maior segurança.

13.5.5 VALOR

O último V é o que torna Big Data relevante: tudo bem ter acesso a uma quantidade massiva de informação a cada segundo, mas isso não adianta nada se não puder gerar valor algum para um órgão ou uma empresa. É importante que organizações entrem no negócio do Big Data, mas é sempre importante lembrar dos custos e benefícios, além de tentar agregar valor ao que se está fazendo. *Bacana?*

O quinto desafio posto pelo Big Data pelas empresas é o de definir a abordagem que será feita dessa massa de dados que está circulando. Afinal, para que um dado se converta em informação útil e utilizável é preciso o olho do analisador, é preciso colocar uma pergunta a esse dado que permita orientar a análise de dados para o objetivo de uma empresa. Não é toda a informação que está circulando que é relevante ou útil para os objetivos específicos de uma empresa.

PREMISSAS	DESCRIÇÃO
VOLUME	Corresponde à grande quantidade de dados acumulada.
VELOCIDADE	Corresponde à rapidez na geração e obtenção de dados.
VARIEDADE	Corresponde à grande diversidade de tipos ou formas de dados.
VERACIDADE	Corresponde à confiança na geração e obtenção dos dados.
VALOR	Corresponde à utilidade e valor agregado ao negócio.

Volume	O volume da informação refere-se ao fato de que certas coleções de dados atingem a faixa de gigabytes (bilhões de bytes), terabytes (trilhões), petabytes (milhares de trilhões) ou mesmo exabytes (milhões de trilhões). Assim, o Big Data deve possibilitar a análise de grandes volumes de dados . Além disso, a tecnologia do Big Data serve exatamente para lidar com esse volume de dados, guardando-os em diferentes localidades e juntando-os através de software .
Velocidade	Está relacionada à rapidez com a qual os dados são produzidos e tratados para atender à demanda , o que significa que não é possível armazená-los por completo, de modo que somos obrigados a escolher dados para guardar e outros para descartar. A tecnologia de Big Data agora nos permite analisar os dados no momento em que estes são gerados, SEM a necessidade de inseri-los nos bancos de dados . Exemplos de uso envolvendo a tomada de decisão em tempo real: detecção de fraude em transação financeira; detecção de doença grave em <i>check-up</i> ; etc.
Variedade	O Big Data deve ser capaz de lidar com diferentes formatos de informação , como, por exemplo, arquivos de texto, <i>e-mail</i> , medidores e sensores de coleta de dados, vídeo, áudio, dados de ações do mercado ou transações financeiras. Dados são gerados em inúmeros formatos — desde estruturados (numéricos, em <i>databases</i> tradicionais) a não estruturados (documentos de texto, <i>e-mail</i> , vídeo, áudio, cotações da bolsa e transações financeiras, etc.).
Veracidade	Quanto à veracidade, Weber <i>et. al.</i> (2009) ressaltou que as informações verdadeiras podem ser usadas pelos gestores para responder aos desafios estratégicos . A veracidade garantiria, então, a confiabilidade dos dados . Não adianta lidar com a combinação “volume + velocidade + variedade” se não houver dados confiáveis. É necessário que haja processos que garantam a consistência dos dados. A veracidade refere-se mais à proveniência ou à confiabilidade da fonte de dados , seu contexto e a sua utilidade para a análise com base nela.
Valor	Os dados do Big Data devem agregar valor ao negócio . O último V, valor, portanto, considera que informação é poder, informação é patrimônio. Com relação ao valor, Chen <i>et. al.</i> (2014) afirmam que as análises críticas de dados podem ajudar as empresas a melhor entender seus negócios trazendo benefícios. A combinação “volume + velocidade + variedade + veracidade”, além de todo e qualquer outro aspecto que caracteriza uma solução de <i>Big Data</i> , será inviável se o resultado não trouxer benefícios significativos e que compensem o investimento.

13.5. CLOUD COMPUTING:



Ter um sistema de computação em nuvem é condição para se trabalhar bem com um grande volume de dados, uma vez que isso envolve coleta, armazenamento e compartilhamento de um número gigantesco de informações.

Além disso, a constante necessidade de conhecer o resultado das ações de um negócio, muitas vezes, imediatamente, torna essa relação entre Cloud Computing e Big Data extremamente harmoniosa.

A computação em nuvem é a infraestrutura geralmente utilizada para suportar iniciativas de Big Data! Como a Cloud Computing possui capacidade para processar grandes volumes de dados em tempo real a tornando praticamente indissociável de Big Data quando o assunto é gerar vantagens competitivas para uma organização a partir das informações que ela possui disponíveis, seja internamente ou no mercado. A grande vantagem de associar Big Data à Cloud Computing é reduzir os custos de uma infraestrutura para armazenar e processar os dados.

13.6. MapReduce

Se esse computador poderoso tem a capacidade de verificar que possa olhar através de um milhão de números por hora, ele precisará – portanto – de 100 horas (+- 4 dias) para verificar todos os cem milhões de números. Agora, se você dividir essa lista em cem partes e as entregar para 100 computadores, cada computador pesquisará em uma lista de apenas 1 milhão de números e poderá encontrar o maior número dessa lista em 1 hora.

Após cada computador encontrar seu maior número, bastam alguns segundos para encontrar o maior dentre esses cem números. Logo, o trabalho que foi realizado em cerca de quatro dias por um único computador poderá ser finalizado em cerca de uma hora pelos cem computadores. O processo de decomposição dos dados é chamado de Mapeamento (Map); e o processo de consolidação do resultado dos mapeamentos é chamado de Redução (Reduce).

O MapReduce é um modelo de programação que permite reduzir problemas grandes em problemas menores, mapeando cada subproblema para máquinas diferentes (ou processadores diferentes de uma mesma máquina) e, em seguida, reduzindo cada resposta intermediária à única resposta final que você está procurando.

MapReduce divide o conjunto de dados de entrada em blocos independentes que são processados pelas tarefas de mapa de uma maneira completamente paralela. Essa estrutura classifica as saídas dos mapas, as quais são, então, inseridas nas tarefas de redução.

O MapReduce é um modelo de programação que permite reduzir problemas grandes em problemas menores, mapeando cada subproblema para máquinas diferentes (ou processadores diferentes de uma mesma máquina) e, em seguida, reduzindo cada resposta intermediária à única resposta final que você está procurando. O processo de decomposição dos dados é chamado de Mapeamento/Mapa (Map); e o processo de consolidação do resultado dos mapeamentos é chamado de Redução (Reduce). De fato, as saídas dos mapas são inseridas como entradas na etapa de redução, sendo essa responsável por consolidar os resultados de cada mapa, gerando um resultado agregado.

Apache Hadoop é apenas uma das implementações da técnica de MapReduce – existem outras implementações, mas essa é a mais famosa! Em outras palavras, ele é um software de código aberto, implementado na linguagem de programação Java, para implementar o algoritmo de MapReduce em máquinas comuns.

13.7. Hadoop

- Quando nos referimos a *Big Data*, apenas um banco de dados do tipo não basta. É necessário também contar com ferramentas (Ex.: Hadoop é a principal referência) que permitam o tratamento correto do volume de dados.

- Hadoop: plataforma *open source* desenvolvida especialmente para processamento e análise de grandes volumes de dados, sejam eles estruturados ou não estruturados. – É utilizado em larga escala por grandes corporações, como Facebook e Twitter, em aplicações Big Data.

- Útil para aplicações que envolvam dados massivos para processamento paralelo (embora seja interessante para processamento de quaisquer dados), geralmente utilizando um cluster de computadores

- Trata-se de um projeto da Apache de alto nível, que vem sendo construído por uma

comunidade de colaboradores utilizando em sua maior parte a linguagem de programação Java, com algum código nativo em C e alguns utilitários de linha de comando escrito utilizando *scripts shell*,

Na verdade, ele é mais do que um software – ele é uma plataforma, um framework, um ecossistema de computação distribuída orientada a clusters e voltado para armazenamento e processamento de grandes volumes de dados, com alta escalabilidade, grande confiabilidade e tolerância a falhas. Ele é responsável por todo gerenciamento do cluster, não sendo necessário configurar máquinas individualmente.

Em implementações de Big Data, temos uma arquitetura baseada em dois componentes principais: armazenamento de dados e processamento paralelo. No Hadoop, o armazenamento distribuído de dados é tratado pelo HDFS (Hadoop File System¹) e o processamento paralelo de dados é tratado pelo MapReduce. Em suma, podemos dizer que o Hadoop é uma combinação do MapReduce e do HDFS (que foi inspirado no GoogleFS – Google FileSystem).

HADOOP: plataforma em Java e voltada para clusters, inspirada no MapReduce e no GoogleFS, de computação distribuída muito utilizada no contexto de Big Data para o processamento de grandes massas de dados. Hadoop é mais do que um software – ele é uma plataforma, um framework, um ecossistema de computação distribuída orientada a clusters e voltado para armazenamento e processamento de grandes volumes de dados, com alta escalabilidade, grande confiabilidade e tolerância a falhas.

13.8. Normalização Big Data

A normalização de dados em Big Data refere-se ao processo de transformar e padronizar dados de diversas fontes e formatos em uma estrutura comum e consistente, facilitando a análise e integração de informações. Esse processo é essencial para garantir que os dados possam ser comparados, analisados e utilizados de forma eficiente em aplicações de Big Data.

A normalização de dados pode incluir várias etapas, como:

- **Limpeza de dados:** Remoção de erros, inconsistências, duplicatas e informações irrelevantes ou incompletas. Isso melhora a qualidade dos dados e reduz o ruído na análise.
- **Transformação de dados:** Conversão dos dados em um formato padrão ou unificado, de modo que possam ser facilmente comparados e analisados. Por exemplo, converter diferentes formatos de data e hora para um formato padrão.
- **Integração de dados:** Consolidação de dados de várias fontes, como bancos de dados, planilhas, arquivos de texto e sistemas de gerenciamento de conteúdo, em um único repositório. Isso permite uma análise mais abrangente e integrada das informações.
- **Redução de dimensionalidade:** Redução do número de variáveis ou atributos nos dados, mantendo apenas aqueles que são relevantes e significativos para a análise. Isso pode ser feito por meio de técnicas de seleção de recursos, análise de componentes principais (PCA) ou outras técnicas de redução de dimensionalidade.
- **Escalonamento e normalização:** Padronização das escalas das variáveis, de modo que todas as características tenham o mesmo peso e importância na análise. Isso pode ser feito por meio de técnicas como min-max scaling, z-score normalization ou outras técnicas de escalonamento e normalização.

Em resumo, a normalização de dados em Big Data é um processo crucial para garantir a qualidade, consistência e utilidade dos dados, permitindo uma análise eficiente e insights valiosos a partir de grandes volumes de informações.

13.9. PIPELINE Big Data

- Um pipeline de dados é uma série de etapas de processamento para preparar dados corporativos para análise.
- As organizações têm um grande volume de dados de várias fontes, como aplicativos, dispositivos de Internet das Coisas (IoT) e outros canais digitais. No entanto, os dados brutos são inúteis; eles devem ser movidos, classificados, filtrados, reformatados e analisados para *business intelligence*.
- Um pipeline de dados inclui várias tecnologias para verificar, resumir e encontrar padrões nos dados para informar as decisões de negócios.
- Pipelines de dados bem organizados oferecem suporte a vários projetos de big data, como visualizações de dados, análises exploratórias de dados e tarefas de machine learning.

13.10. TIPOS DE VARIÁVEIS

13.11.1 Variável DEPENDENTE

Em big data, a variável dependente é um termo usado para descrever uma variável cujo valor é baseado ou influenciado por uma ou mais outras variáveis, chamadas de variáveis independentes. No contexto de big data, as variáveis dependentes são geralmente usadas em análises estatísticas e modelagem preditiva para identificar padrões e fazer previsões.

Em um conjunto de dados, a variável dependente é aquela que você deseja prever ou explicar, enquanto as variáveis independentes são aquelas que você usa para fazer a previsão. Por exemplo, em um estudo sobre o impacto da publicidade na receita de uma empresa, a receita seria a variável dependente (aquela que você deseja prever), enquanto o gasto com publicidade seria uma das variáveis independentes (usadas para fazer a previsão).

13.11.2 Variável INDEPENDENTE

Em big data, as variáveis independentes são aquelas que influenciam ou afetam a variável dependente. Elas são chamadas de "independentes" porque seus valores não dependem de outras variáveis no conjunto de dados. As variáveis independentes são usadas para explicar, prever ou estabelecer uma relação com a variável dependente.

Em análises estatísticas e modelagem preditiva, as variáveis independentes são tratadas como entradas ou fatores que contribuem para a previsão da variável dependente. No exemplo anterior sobre o impacto da publicidade na receita de uma empresa, o gasto com publicidade seria uma variável independente. Outras variáveis independentes poderiam incluir fatores como preço dos produtos, localização das lojas, e atividades promocionais.

É importante notar que, embora uma variável possa ser classificada como independente em um contexto específico, ela pode ser dependente em outro. A classificação de uma variável como dependente ou independente depende da relação que está sendo estudada e do objetivo da análise.

13.11. O gerenciamento de Big Data

envolve o processo de coleta, armazenamento, processamento e análise de grandes volumes de dados estruturados e não estruturados, provenientes de diversas fontes e em alta velocidade. O objetivo do gerenciamento de Big Data é extrair informações valiosas e insights que possam auxiliar na tomada de decisões e na obtenção de vantagens competitivas.

O gerenciamento de Big Data abrange várias etapas e desafios:

- **Coleta de dados:**
Envolve a identificação e aquisição de dados relevantes a partir de várias fontes, como sensores, dispositivos móveis, redes sociais, logs e transações comerciais.
- **Armazenamento de dados:**
Devido ao grande volume de dados gerados, é crucial selecionar as soluções de armazenamento adequadas, como data warehouses, data lakes ou sistemas de armazenamento distribuídos, como o Hadoop Distributed File System (HDFS).
- **Processamento de dados:**
Os dados devem ser processados para serem analisados e transformados em informações úteis. Isso pode incluir a limpeza e a padronização de dados, bem como a aplicação de algoritmos e técnicas de aprendizado de máquina para análise.
- **Análise de dados:**
A análise de Big Data pode envolver técnicas de mineração de dados, aprendizado de máquina, análise de texto, análise de sentimentos e análise de redes sociais, para identificar padrões, tendências e correlações que possam fornecer insights valiosos.
- **Visualização de dados:**
A apresentação dos resultados da análise de Big Data de forma visual facilita a compreensão e a interpretação das informações por parte dos tomadores de decisão. Ferramentas de visualização de dados, como dashboards e gráficos, são essenciais para comunicar os insights extraídos.
- **Segurança e privacidade:**
O gerenciamento de Big Data também envolve a proteção dos dados contra acesso não autorizado, violações de segurança e vazamentos de informações. Além disso, é

necessário garantir a conformidade com as leis e regulamentações de privacidade de dados, como o GDPR e a LGPD.

- **Governança de dados:**

A governança de dados é um aspecto crítico do gerenciamento de Big Data, pois estabelece políticas, procedimentos e padrões para garantir a qualidade, integridade, consistência e conformidade dos dados ao longo de todo o ciclo de vida dos mesmos.

Em resumo, o gerenciamento de Big Data é um processo complexo que envolve a coleta, armazenamento, processamento, análise e visualização de grandes volumes de dados, com o objetivo de extrair informações valiosas e insights. A eficácia do gerenciamento de Big Data depende da seleção das soluções tecnológicas apropriadas e da implementação de práticas sólidas de segurança, privacidade e governança de dados.

O Big Data analítico refere-se ao processo de examinar e analisar grandes volumes de dados estruturados e não estruturados com o objetivo de descobrir informações úteis, identificar padrões e tendências ocultas, e gerar insights valiosos para a tomada de decisões. Essa abordagem analítica permite que as organizações tomem decisões informadas, otimizem processos, melhorem a eficiência e obtenham vantagens competitivas.

13.12. Big Data analítico

Envolve a aplicação de diversas técnicas e ferramentas, incluindo:

- **Mineração de dados:**

Utiliza algoritmos e técnicas estatísticas para identificar padrões e correlações nos dados, ajudando a prever comportamentos e tendências futuras.

- **Aprendizado de máquina:**

Emprega algoritmos e modelos matemáticos para analisar grandes volumes de dados e aprender com eles, melhorando automaticamente seu desempenho na identificação de padrões e na tomada de decisões.

- **Análise de texto:**

Envolve o processamento e a análise de dados não estruturados, como textos, e-mails e documentos, para extrair informações relevantes e identificar padrões e sentimentos.

- **Análise de redes sociais:**

Examina as interações e conexões em redes sociais para identificar tendências e influenciadores, bem como compreender o comportamento e as preferências dos usuários.

- **Visualização de dados:**

Transforma os resultados da análise de Big Data em gráficos, tabelas e painéis interativos, facilitando a interpretação e a comunicação das informações e insights gerados.

Além das técnicas e ferramentas mencionadas, o sucesso do Big Data analítico depende de uma infraestrutura adequada para armazenar, processar e analisar os dados em alta velocidade e em grande escala. Tecnologias como Hadoop, Spark e NoSQL são amplamente utilizadas para lidar com os desafios do Big Data analítico.

Em resumo, o Big Data analítico é uma abordagem que permite às organizações extrair insights valiosos a partir de grandes volumes de dados estruturados e não estruturados. A aplicação de técnicas avançadas de análise e a utilização de ferramentas e tecnologias apropriadas são

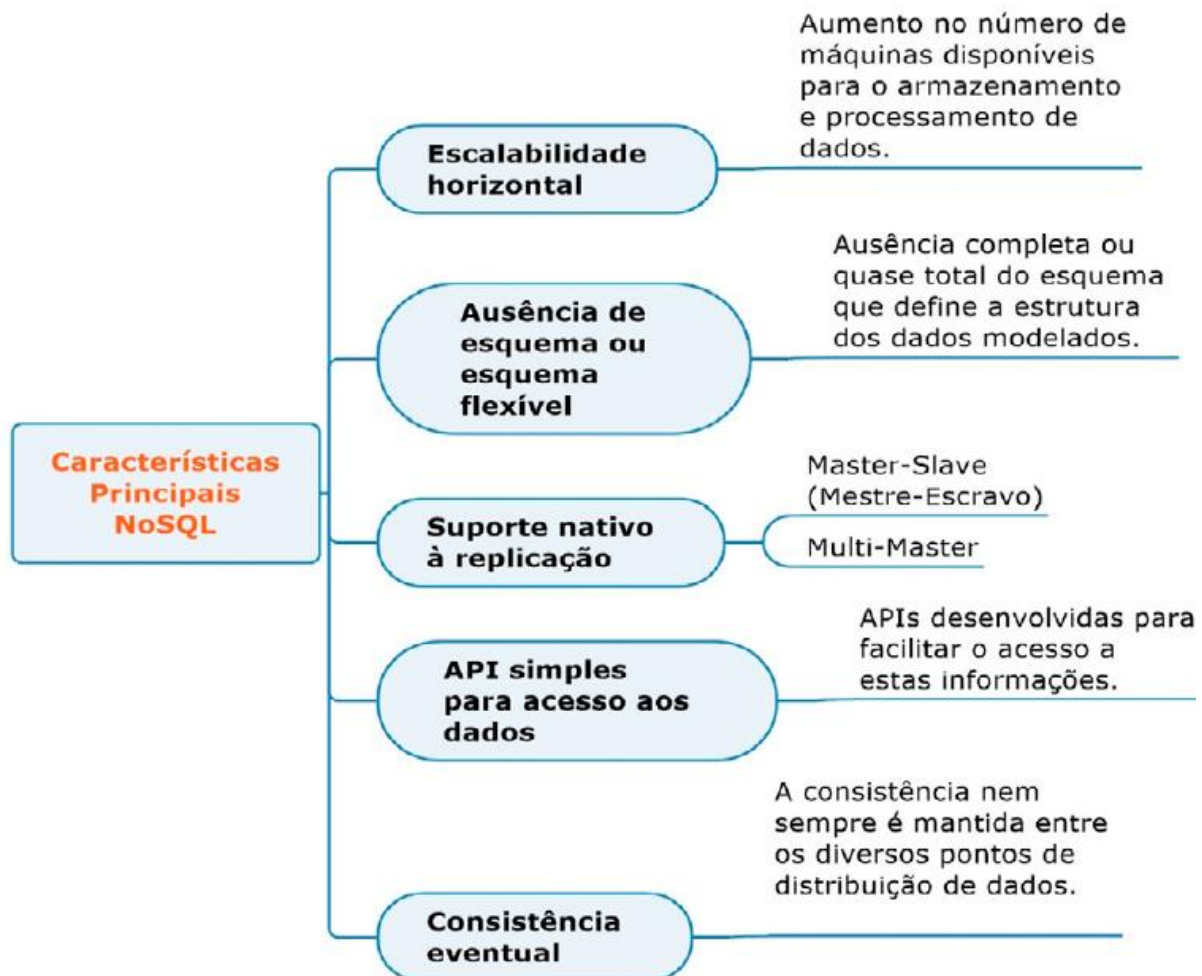
fundamentais para transformar esses dados em informações úteis, que podem ser usadas para impulsionar a tomada de decisões e melhorar o desempenho organizacional.

14.NoSQL (Not only SQL)

Esse tipo de estrutura de dados visa possibilitar o alto armazenamento através de velocidade e grande disponibilidade.

CRITÉRIO	NOSQL	SQL
MODELO	Não-Relacional	Relacional
ARMAZENAMENTO	Variados (Grafos, Documentos, etc)	Tabelas
FLEXIBILIDADE	Alta flexibilidade (Esquema indefinido)	Baixa flexibilidade (Esquema definido)
ADEQUAÇÃO	Mais adequado a dados não-estruturados	Mais adequado a dados estruturados
ESCALABILIDADE	Em geral, escalabilidade horizontal	Em geral, escalabilidade vertical
SGBD	MongoDB, Cassandra, HBase, Neo4J, etc	Oracle, MySQL, DB2, SQL Server, etc

TIPO DE MODELOS	DESCRIÇÃO
ORIENTADO A CHAVE-VALOR	Esse modelo armazena dados por meio de uma estrutura de mapeamento ou dicionário, em que todo dado armazenado possui uma chave identificadora e seu valor em si. Para cada chave de entrada, é retornado um valor de saída (Ex: Table Storage, DynamoDB, Cassandra e Redis).
ORIENTADO A DOCUMENTOS	Esse modelo armazena dados na forma de documentos flexíveis, semiestruturados e hierárquicos junto com seus metadados sem uma estrutura definida. Em geral, os dados são armazenados em formato JSON ou XML (Ex: MongoDB, CouchDB e DocumentDB).
ORIENTADO A GRAFOS	Esse modelo armazena o relacionamento entre dados altamente conectados por meio de vértices e arestas. São geralmente utilizados em redes sociais, mecanismos de recomendação e detecção de fraudes (Ex: Neo4J, Infinite Graph e ArangoDB).
ORIENTADO A COLUNAS	Esse modelo armazena dados em colunas dinâmicas. É o mais semelhante ao modelo relacional, mas os dados são armazenados em colunas em vez de linhas. Ademais, cada coluna pode conter subcolunas, que podem conter várias propriedades (Ex: Hypertable e MonetDB).



Obs.: Os bancos de dados não relacionais (NoSQL) não utilizam o esquema tradicional de tabela de linhas e colunas; em vez disso, eles usam um modelo de armazenamento otimizado para desempenho escalável e modelos de dados sem esquema.

BASE:

- *Basically Available,*
- *Soft state,*
- *Eventually consistency*

Distribui os dados em diferentes repositórios tornando-os sempre disponíveis, não se preocupa com a consistência de uma transação, delegando essa função para a aplicação, porém sempre garante a consistência dos dados em algum momento futuro à transação.

O BASE pode ser explicado em contraste com outra filosofia de *design* – Atomicidade, Consistência, Isolamento, Durabilidade (ACID). O modelo ACID promove consistência sobre disponibilidade, enquanto o BASE promove disponibilidade sobre consistência.

ACID.

- Atomicidade
- Consistência
- Isolamento.
- Durabilidade

- **ATOMICIDADE** visa garantir que uma transação é uma unidade atômica de processamento (indivisível). Logo, a transação será executada em sua totalidade, com todas as suas operações finalizadas e refletidas no BD, ou não será executada de modo algum, e nenhuma das suas operações são refletidas no BD.
- **CONSISTÊNCIA**, a execução completa da transação deverá levar o banco de dados de um estado consistente para outro estado consistente, onde um estado consistente do banco de dados satisfaz as restrições especificadas no esquema, bem como quaisquer outras restrições que devam controlar o banco de dados.
- **ISOLAMENTO** determina que uma transação deve ser executada como se estivesse isolada das demais. Cada transação assume que está sendo executada sozinha no sistema. O sistema garante que os resultados intermediários da transação permaneçam escondidos de outras transações executando concorrentemente.
- **DURABILIDADE** busca assegurar que as mudanças aplicadas no banco de dados por uma transação efetivada devem persistir no banco de dados. Estas mudanças não devem ser perdidas em razão de uma falha.

14.1. Teorema CAP (ou Teorema de Brewer)

O Teorema CAP (*Consistency, Availability e Partition tolerance*) é aplicado aos bancos de dados não relacionais. Segundo esse teorema, em um sistema distribuído, em um dado momento, só é possível garantir duas destas três propriedades, que seriam consistência, disponibilidade e tolerância à partição.

a) Consistência (*Consistency*)

Permite que todos os clientes veem os mesmos dados ao mesmo tempo, não importa em qual nó eles se conectem.

Para que isso aconteça, sempre que os dados forem gravados em um nó, ele deve ser instantaneamente encaminhado ou replicado para todos os outros nós do sistema antes que a gravação seja considerada "bem-sucedida".

b) Disponibilidade (*Availability*)

Qualquer cliente que fizer uma solicitação de dados obterá uma resposta, mesmo que um ou mais nós estejam desativados. Dessa forma, todos os nós em funcionamento no sistema distribuído retornam uma resposta válida para qualquer solicitação, sem exceção.

c) Tolerância de partição (*Partition Tolerance*)

A partição é uma quebra de comunicações dentro de um sistema distribuído, uma conexão perdida ou temporariamente lenta entre dois nós.

Tolerância de partição significa que o cluster deve continuar a funcionar mesmo de ocorrer uma ou mais falhas de comunicação entre os nós no sistema.

14.2. Principais Características dos Bancos de Dados NoSQL

- Ausência de esquema (Schema-free) ou esquema flexível

As tecnologias NoSQL foram desenvolvidas para suportar aplicações que manipulam grandes volumes de dados, e que necessitam de um modelo de dados mais flexível, sem esquema fixo (schemaless).

- Escalabilidade horizontal

Uma importante característica sobre os bancos *NoSQL*, é que eles foram projetados para serem executados em *clusters* de computadores favorecendo a escalabilidade horizontal, na qual o modelo relacional é incompatível

A diferença fundamental entre escalabilidade horizontal e vertical é que a primeira se concentra na adição de mais máquinas para distribuir a carga de trabalho, enquanto a segunda se concentra em adicionar mais recursos em uma única máquina para aumentar sua capacidade de lidar com a carga de trabalho.

- Outra característica particular de bancos NoSQL é que nem sempre a consistência dos dados é mantida.

- Bancos de Dados NoSQL são ideais para aplicações de *Big Data*.

- Bancos de Dados NoSQL oferecem suporte nativo à replicação: esta é outra forma de prover a escalabilidade, pois, no momento em que permitimos a replicação de forma nativa o tempo gasto para recuperar informações é reduzido.

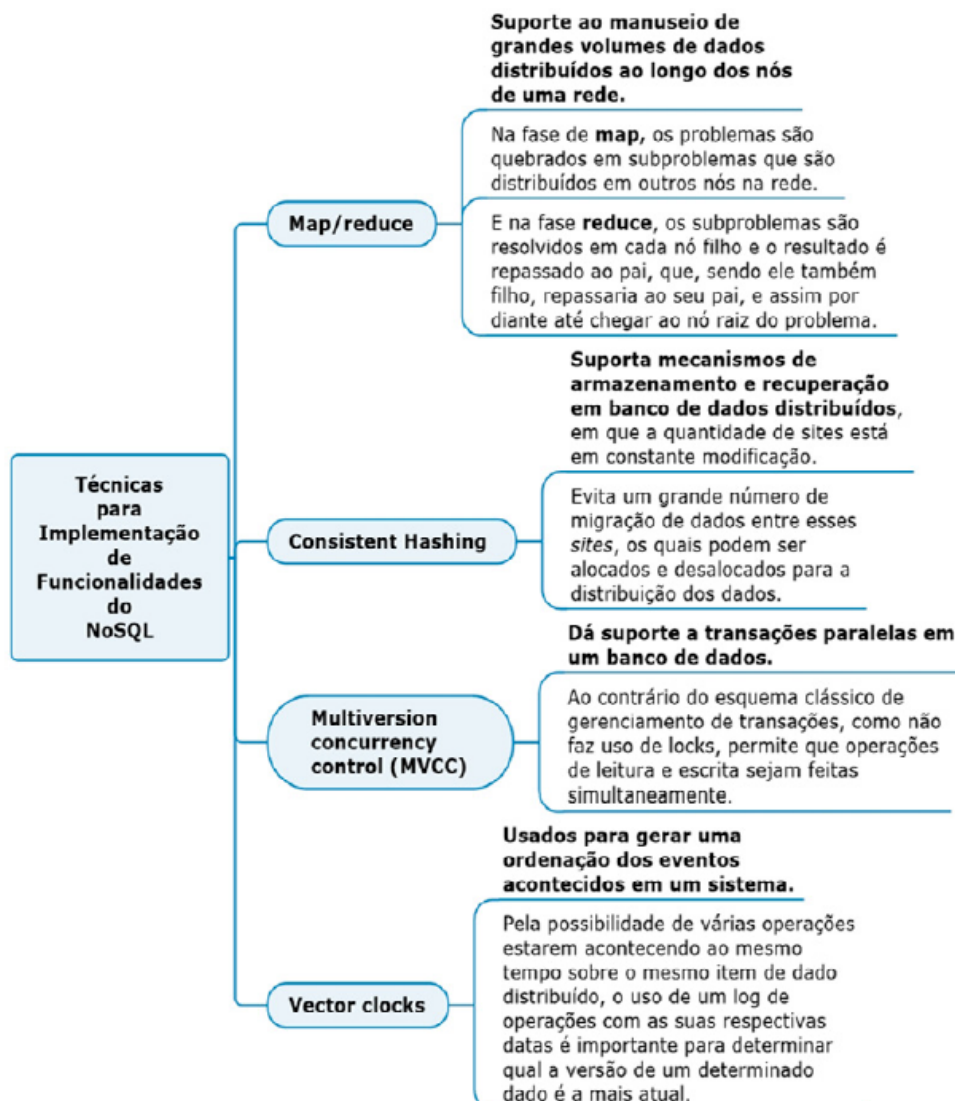


Figura. Técnicas para Implementação de Funcionalidades NoSQL

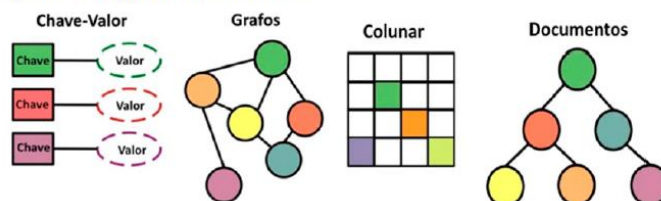
Use Bancos de Dados Relacionais quando...	Use NoSQL quando...
Suas aplicações forem centralizadas (ERP, CRM).	Suas aplicações forem descentralizadas (Web, Mobile, Big Data, IoT)
Alta disponibilidade moderada for necessária.	Quando a disponibilidade tiver que ser contínua, sem interrupção.
Dados gerados em velocidade moderada.	Dados gerados em alta velocidade (sensores).
Dados forem gerados a partir de poucas fontes.	Dados forem gerados a partir de múltiplas fontes.
Dados forem estruturados.	Dados forem semiestruturados ou não estruturados.
Transações complexas.	Transações simples.
For necessário manter moderado volume de dados.	For necessário manter alto volume de dados.

Use Bancos de Dados Relacionais quando...	Use NoSQL quando...
Dados são armazenados com base no modelo relacional.	Dados são armazenados em <i>clusters</i> de computadores.
Utilizam JOINS, tornando-os uma boa escolha para aplicações que envolvem a gestão de várias operações (PEREIRA, 2016).	A utilização de JOINS é <i>limitada</i> .
Possuem esquema fixo (tipos de dados sobre colunas).	Possuem esquema dinâmico, muitas vezes referenciado como <i>schemaless</i> .
Suportam escala vertical.	Suportam escala horizontal.
Compatível com propriedades ACID.	Alguns bancos NoSQL suportam transações ACID.
A manipulação dos dados é via linguagem SQL.	A manipulação dos dados é via APIs próprias da tecnologia ou através de um subconjunto limitado da linguagem SQL.

	Modelo de Dados Relacional	Modelo de Dados NoSQL
Escalonamento	Possível, mas complexo . Devido à natureza estruturada do modelo, a adição de forma dinâmica e transparente de novos nós no grid não é realizada de modo natural.	Uma das principais vantagens desse modelo. Por não possuir nenhum tipo de esquema predefinido, o modelo possui maior flexibilidade o que favorece a inclusão transparente de outros elementos.
Consistência	Ponto mais forte do modelo relacional. As regras de consistência presentes propiciam um maior grau de rigor quanto à consistência das informações.	Realizada de modo eventual no modelo: só garante que, se nenhuma atualização for realizada sobre o item de dados, todos os acessos a esse item devolverão o último valor atualizado.
Disponibilidade	Dada a dificuldade de se conseguir trabalhar de forma eficiente com a distribuição dos dados, esse modelo pode não suportar a demanda muito grande de informações do banco.	Outro fator fundamental do sucesso desse modelo. O alto grau de distribuição dos dados propicia que um maior número de solicitações aos dados seja atendida por parte do sistema e que o sistema fique menos tempo não disponível.

PRINCIPAIS MODELOS DE BANCO DE DADOS NoSQL

A seguir, destacamos os principais **modelos de gerenciamento de dados NoSQL**. Existem quatro grandes **categorias** mais utilizadas:



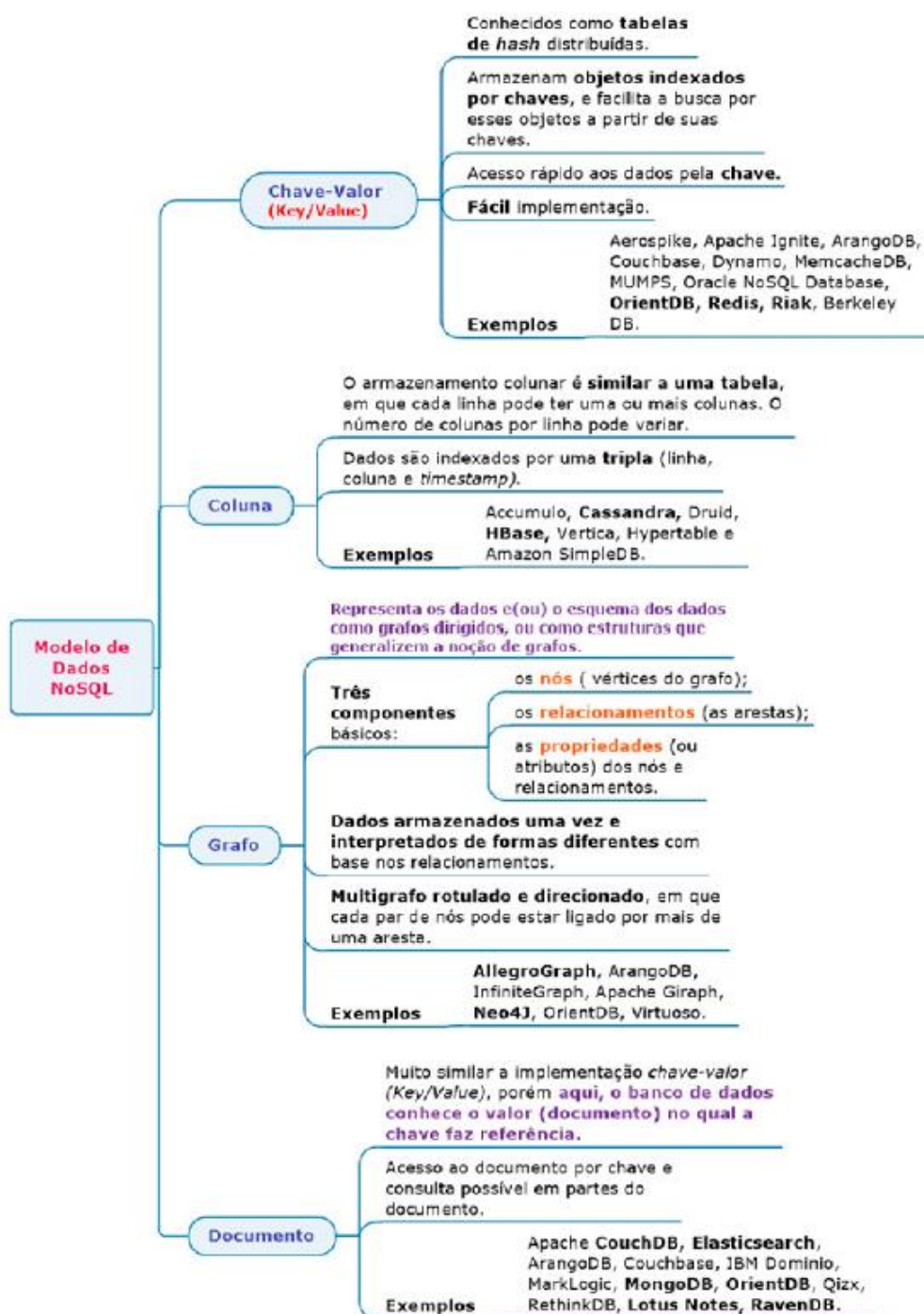


Figura. Principais Modelos de Dados NoSQL

14.3. Modelo NoSQL Chave-Valor (Key-Value)

Key/Value Store é o tipo de banco de dados NoSQL mais simples, cujo conceito é uma chave e um valor para essa chave. No entanto, ele é o que aguenta maior carga de dados e possui a maior escalabilidade.

Sistemas distribuídos nessa categoria, também conhecidos como tabelas de *hash* distribuídas, armazenam objetos indexados por chaves e possibilitam a busca por esses objetos a partir de suas chaves.

14.4. Modelo NoSQL Orientado a Documentos

Muito similar a implementação *chave-valor (Key/Value)*, porém aqui, o banco de dados conhece o valor (documento) no qual a chave faz referência. O banco de dados *NoSQL* entende o valor armazenado, permitindo realizar interações sobre ele.

Em geral, os bancos de dados orientados a documentos não possuem esquema, ou seja, os documentos armazenados não precisam possuir estrutura em comum.

14.5. Modelo Orientado a Colunas (ou Colunar)

O armazenamento colunar é similar a uma tabela, em que cada linha pode ter uma ou mais colunas. O número de colunas por linha pode variar.

É possível pensar em um armazenamento de dados de família de colunas como dados contidos em tabela com linhas e colunas, mas as colunas são divididas em grupos conhecidos como famílias de colunas.

Cada família de colunas contém um conjunto de colunas que estão logicamente relacionadas e geralmente são recuperadas ou manipuladas como uma unidade. Outros dados acessados separadamente podem ser armazenados em famílias de colunas separadas

	OLAP (On-line Analytical Processing)	OLTP (On-line Transaction Processing)
Foco	Foco no nível estratégico da organização. Visa a ANÁLISE empresarial e tomada de decisão.	Foco no nível operacional da organização. Visa a execução operacional do negócio.
Operação Típica	Análise.	Atualização.
Performance	Otimização para a leitura e geração de análises e relatórios gerenciais.	Alta velocidade na manipulação de dados operacionais, porém ineficiente para geração de análises gerenciais.
Estrutura dos dados	Os dados estão estruturados na modelagem multidimensional. Os dados normalmente possuem alto nível de sumarização.	Os dados são normalmente estruturados em um modelo relacional normalizado, otimizado para a utilização transacional. Os dados possuem alto nível de detalhes.
Armazenamento	Feito em estruturas de Data Warehouse com otimização no desempenho em grandes volumes de dados. É comum o armazenamento maior de dados para OLAP, em relação a OLTP, com a finalidade de se manter histórico para análise.	Feito em sistemas convencionais de Banco de Dados através dos sistemas de informações da organização.
Abrangência	Utilizado pelos gestores e analistas para a tomada de decisão.	Utilizado por técnicos e analistas. Engloba também vários usuários da organização.

	OLAP (On-line Analytical Processing)	OLTP (On-line Transaction Processing)
Frequência de atualização	A atualização das informações é feita no processo de carga dos dados. Frequência baixa, podendo ser diária, semanal, mensal ou anual (ou critério específico).	Atualização dos dados é feita no momento da transação. Frequência muito alta de atualizações.
Volatilidade	Dados históricos e não voláteis. Os dados não sofrem alterações, salvo necessidades específicas (por motivos de erros ou inconsistências de informações).	Dados voláteis, passíveis de modificação e exclusão.
Tela	Definida pelo usuário	Imutável
Nível de Dados	Altamente sumarizado	Atomizado
Idade dos Dados	Histórico, atual, projetado	Presente
Recuperação	Muitos Registros	Poucos Registros

14.6. Modelo NoSQL Orientado a Grafos

Semelhante ao modelo relacional, aqui é mais explícito o relacionamento entre os dados. Muito útil para dados que estejam bastante interligados, pois a estrutura de um grafo expõe naturalmente os relacionamentos entre os objetos (ARMBRUST, 2021).

Diferentemente de outros tipos de bancos de dados NoSQL, esse está diretamente relacionado a um modelo de dados estabelecido, o modelo de grafos.

A ideia desse modelo é representar os dados e(ou) o esquema dos dados como grafos dirigidos, ou como estruturas que generalizem a noção de grafos.

O modelo de grafos é mais interessante que outros quando informações sobre a interconectividade ou a topologia dos dados são mais importantes ou tão importantes quanto os dados propriamente ditos.

O modelo orientado a grafos possui três componentes básicos:

- os nós (são os vértices do grafo),
- os relacionamentos (são as arestas) e
- as propriedades (ou atributos) dos nós e relacionamentos.

▪ CYPHER QUERY LANGUAGE:

Cypher é uma linguagem de consulta usada para acessar e manipular grafos em bancos de dados orientados a grafos. Foi desenvolvida pela empresa Neo4j e é a linguagem padrão usada em seu banco de dados de grafo.

As consultas Cypher são divididas em duas partes: a cláusula MATCH, que define os padrões do grafo que devem ser encontrados, e a cláusula RETURN, que define as informações que devem ser retornadas como resultado da consulta. Além disso, existem outras cláusulas que podem ser usadas para filtrar, ordenar e limitar os resultados da consulta.

Em resumo, Cypher é uma linguagem de consulta para bancos de dados de grafos, que é projetada para ser simples e expressiva. Ela é amplamente utilizada em aplicações que envolvem dados altamente conectados, permitindo que os usuários realizem consultas complexas em grafos com facilidade e eficiência.

- (CASSANDRA) Colunas como modelo de dados.
- (MONGODB) Documentos como modelo de dados.
- (REDIS) Chave-valor como modelo de dados.

Modelo Estrutural	Principais Fabricantes
Chave-Valor	Riak, Redis, Amazon DynamoDB, Oracle BerkeleyDB
Documento	MongoDB, CouchDB, AzureDocumentDB
Famílias de colunas ou colunar	HBase, Cassandra, HyperTable, Big Table
Grafos	Neo4J, Infinite Graph, InfoGrid, Titan

O ataque de injeção SQL consiste em:

um ataque no qual um código mal-intencionado é inserido em cadeias de caracteres que são passadas posteriormente para uma instância do SQL Server para análise e execução. Qualquer procedimento que construa instruções SQL deve ser verificado quanto a vulnerabilidades de injeção porque o SQL Server executará todas as consultas sintaticamente válidas que receber.

Apesar da característica, os NoSQL estão sujeitos à injeções como qualquer outro banco de dados SQL, dado que o referido SQL Server reconhecerá as instruções e executará, gerando problemas de segurança.