INSTITUTO DE GESTÃO E TECNOLOGIA
DA INFORMAÇÃO

Fundamentos em Engenharia de Dados

Jéssica Coelho

2022

Fundamentos em Engenharia de Dados

Jéssica Coelho

Sumário

Capítulo 1. Capítulo 1: Conceitos Fundamentais da Engenharia de Dados: 4

1.1. Dados, informações, conhecimento e inteligência 4

1.2. Tipos de dados e suas fontes 7

1.3. Big Data 8

1.4. Diferentes tipos de análise: descritiva, diagnóstica, preditiva e prescritiva..... . 10

1.5. O que faz um Engenheiro de Dados? 11

Capítulo 2. Modelagem..... 13

2.1. Modelagem Conceitual 13

2.2. Modelo Lógico 23

2.3. Bancos Relacionais e SGBDs 28

2.4. Modelo físico 30

Capítulo 3. SQL..... 33

3.1. DDL..... 37

3.2. DML..... 38

3.3 DQL 39

3.4. DCL 60

Capítulo 4. Modelagem dimensional 62

4.1. Granularidade 67

4.2. Data Marts..... 69

4.3. Star schema vs Snowflake..... 69

4.4. ETL..... 74

4.5. OLAP (Processamento Analítico On-line) 74

4.6. Arquiteturas OLAP 75

Referências..... 82

Capítulo 1. Conceitos Fundamentais da Engenharia de Dados

Empresas já são orientadas por dados a bastante tempo, buscando entender o que houve, quanto foi vendido, como e para quem, portanto é com base em dados que as decisões estratégicas são tomadas. Em um primeiro momento as fontes eram mais previsíveis e seguiam o padrão conhecido como Relacional, com tabelas, colunas e linhas e em quantidade facilmente administrável com as tecnologias vigentes.

Mas com o fato do mercado ser altamente competitivo e ancorado no advento da tecnologia, novos tipos de dados estavam disponíveis em variados formatos e as empresas entendendo que estes poderiam ser úteis, passaram a coletá-los, mas ainda sem saber ao certo como consultá-los ou utilizá-los. Então com o contínuo avanço da tecnologia, nos últimos anos o poder de armazenamento e principalmente de processamento permitiu que esses "novos" dados, ainda pouco explorados, passassem a ser lidos e insights poderosos pudessem ser criados, direcionando todo o mundo para esse caminho.

Esse cenário demandou a criação de novos profissionais, com novas skills para trabalhar com grandes massas de dados, em formatos completamente diversos, e entregar visões rapidamente. É neste contexto que surgiu o Engenheiro de Dados, o profissional responsável por coletar os dados disponíveis dentro ou fora da organização, em diferentes formatos, Excel, JSON, API, XML, tratá-los, integrá-los, catalogá-los e disponibilizá-los para os demais níveis gerarem análises de negócio.

Para nos tornarmos Engenheiros de Dados eficientes, capazes de construir soluções escaláveis e seguindo boas práticas, é importante nos atentarmos aos conceitos e aos fundamentos que serão tema desse primeiro módulo: Fundamentos em Engenharia de dados.

1.1. Dados, informações, conhecimento e inteligência

Então vamos começar pelos conceitos mais importantes:

DADO: dados são valores brutos, literais, observações documentadas sobre o estado do mundo ou resultado de medição, não contextualizados. “São fatos conhecidos que podem ser registrados e não possuem significado completo. Por exemplo, nome, idade, cor, endereços...” (ELMASRI, NAVATHE, 2011).

Na Figura abaixo temos uma sequência de dados: nome de dois estados, os números 2000 e 3000 e os anos 2020 e 2022. Com base neles podemos inferir algo? Qual a relação entre eles? O que querem nos dizer? Não conseguimos saber muito, a menos que tenhamos o contexto.

Figura 1 – Dados.

São Paulo, 2000, 2020
Minas Gerais,3000,2022

Informação: é exatamente esse contexto que traz significado. Pois quando um conjunto de dados é agrupado, correlacionado, processado, contextualizado e interpretado, conseguimos entender a mensagem que o dado quer nos mostrar e passamos a chamar de informação. Como HENRIQUE, Kassio (2021) explica em seu artigo: “Quando um conjunto de dados é organizado de modo a passar uma mensagem dentro de um contexto, temos informação”.

Portanto utilizando o mesmo grupo de dados do exemplo anterior (Figura 1), se os organizarmos, identificarmos e correlacionarmos, conseguimos entender que o Estado de Minas Gerais em 2020 registrou 2000 casos da Doença X. Já São Paulo para a mesma doença em 2020 registrou 3000.

Figura 2 – Informação.

Estado	Casos de doença X	Ano
Minas Gerais	2000	2020
São Paulo	3000	2022

Conhecimento: o conhecimento é gerado a partir da habilidade de analisar um conjunto de informações correlacionadas e gerar novas informações mais aprofundadas, ou seja, trazer novos fatos úteis atrelados à experiência.

Ainda utilizando o exemplo das Figuras 1 e 2, podemos analisar que São Paulo teve, em 2022, 50% mais casos da doença X registrados do que Minas em 2020.

Inteligência: alguns autores falam ainda sobre a inteligência, que é a capacidade de avaliar os conhecimentos adquiridos sobre um determinado grupo de informações e tomar decisões sobre eles, podendo utilizar outras variáveis e até a experiência adquirida. Seriam os chamados insights.

Com base no conhecimento de que em São Paulo temos maior número da incidência da doença X, como mostrado no tópico anterior, e considerando a época do ano que tem temperatura mais baixa, por exemplo (aqui estou cruzando com outras evidências), podemos entender que a vacinação precisa ser priorizada em São Paulo, ou seja, tomar uma decisão ou ação com base nos dados é chamado de Inteligência.

Dessa maneira temos a seguinte relação entre os conceitos apresentados:

Figura 3 – Comparação Dados, Informação e Conhecimento.

Dados	Informação	Conhecimento
Simples observações sobre o estado do mundo ▪ Facilmente estruturado ▪ Facilmente obtido por máquinas ▪ Freqüentemente quantificado ▪ Facilmente transferível	Dados dotados de relevância e propósito ▪ Requer unidade de análise ▪ Exige consenso em relação ao significado ▪ Exige necessariamente a mediação humana	Informação valiosa da mente humana. Inclui reflexão, síntese, contexto ▪ De difícil estruturação ▪ De difícil captura em máquinas ▪ Freqüentemente tácito ▪ De difícil transferência

Fonte: DAVENPORT, T. & PRUSAK, L

1.2. Tipos de dados e suas fontes

Os dados podem assumir diferentes formatos ou tipos conforme suas fontes, e precisamos entender suas características de modo a definir a melhor estratégica técnica para processá-los. Os dados podem ser:

Estruturados: dados organizados segundo formato pré-estabelecido. Podemos citar como exemplo as tabelas, com suas linhas e colunas, e estas tabelas podem se relacionar entre si através de chaves. O dado estruturado pode estar armazenado em planilhas de Excel, ou algum outro SGBD relacional como SQL Server, MySQL, Oracle e outros.

Semiestruturados: como o nome indica são dados que não seguem o modelo de tabelas, com colunas e linhas em formato rígido predeterminado, mas possuem algum nível de formato, por exemplo utilizando tags ou chave valor.

Exemplo: XML, HTML, JSON.

Não estruturados: dados que não seguem nenhuma organização ou hierarquia predeterminada. Exemplo: foto, vídeo, áudio, postagem em rede social, localização geográfica, IoT (internet das coisas), e-mails, dentre outros.

Figura 4 – Exemplo visual da estrutura dos diferentes tipos de dados.

Fonte: DAVENPORT, T. & PRUSAK, L.

Figura 5 – Comparação de tipos de dados.

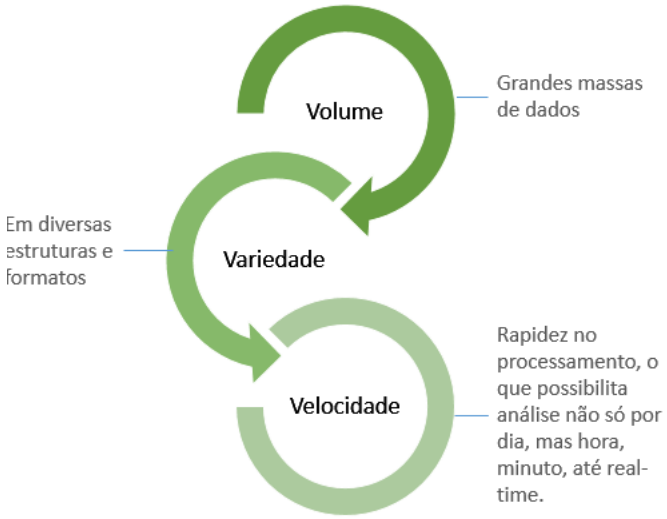
1.3. Big Data

Outro conceito importante na Engenharia de Dados é o Big Data. Para dar um contexto histórico, o termo Big Data foi usado pela primeira vez por dois pesquisadores da NASA, Michael Cox e David Ellsworth, em 1990, quando durante simulações e estudos do fluxo de ar de uma aeronave em voo, se depararam com uma imensa massa de dados gerada pelos super-computadores que não podiam ser processadas ou visualizadas por sobrecarregarem todos os sistemas computacionais. Eles então escreveram no relatório: “Isso é um problema de Big Data”. (URI, Friedman (2012)).

Desde 1990 a quantidade de dados disponíveis e coletadas, em formatos cada vez mais diversos, aumentou exponencialmente, assim como o poder da tecnologia, tornando possível o processamento de tal massa de dados e com isso Big Data se tornou um termo comum para quem trabalha com dados.

Mas nem toda grande massa de dados é considerada Big Data. Isso porque Big Data é caracterizado pelos chamados V’s, sendo Volume, Variedade e Velocidade os três principais. Além deles alguns autores ainda falam que Veracidade e Valor compõem o grupo dos Vs mais importantes.

Figura 6 – Os 3 principais V’s do Big Data.



O volume se refere à quantidade de dados produzidos e coletados pelas organizações. Estes dados podem ser estruturados, semiestruturados ou não estruturados, conferindo variedade. A velocidade por sua vez nos diz em quanto tempo os dados passarão por todas as etapas de tratamento e organização para serem disponibilizados para análises estratégicas e tomada de decisão. Hoje muitas visões já são real-time. Além desses principais, é importante que o dado seja confiável e verdadeiro, ou verídico, e gere valor, pois caso contrário não valeria o investimento.

Além destes 5 alguns autores podem apontar 7, 10 ou até mais. Para fins didáticos vamos focar nos 5 mencionados.

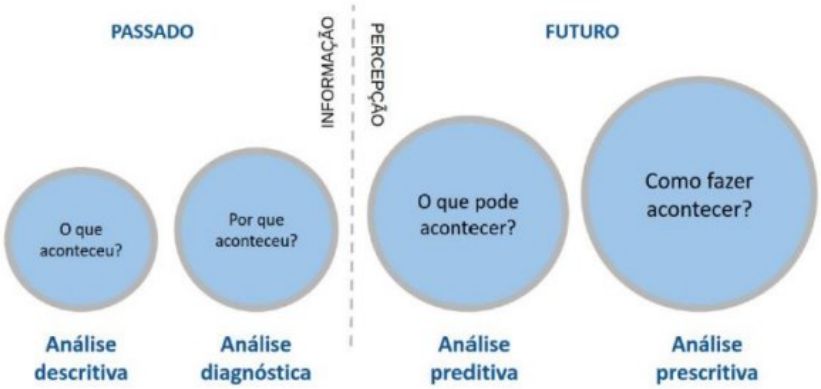
1.4. Diferentes tipos de análise: descritiva, diagnóstica, preditiva e prescritiva

Após coletar, modelar e organizar os dados, gerando informação, são feitas as análises. Nesta fase, de acordo com (DAVENPORT, 2014), são empregadas técnicas para transformação destas informações em conhecimento sobre o cenário avaliado, a fim de encontrar padrões, compreender comportamento de pessoas, produtos, mercado e com isso identificar novas oportunidades de negócio ou melhorar o relacionamento com aqueles que interagem com o seu negócio.

Anteriormente, em função dos tipos de dados coletados e das tecnologias existentes, só era possível uma análise sobre o passado, sobre o que aconteceu, quanto vendi, como e para quem, e com base nisso inferir o futuro. Mais tarde, com o mercado cada vez mais competitivo não era suficiente para as empresas ter em mãos os dados antigos para planejar o futuro, era preciso prever com maior precisão possível o que ia acontecer, qual a tendência, para que elas se antecipassem. E como existe o limite que a mente humana consegue visualizar com clareza sobre o futuro, foi necessário criar meios automáticos, utilizando o máximo de dados sobre o negócio que se poderia coletar, de diversas fontes, aliado à estatística para construir algo assertivo. Desta forma

surgiram os diferentes níveis de análise: descritivas, diagnósticas, preditiva ou prescritiva.

Figura 7 – Diferentes tipos de análise de dados.



Fonte: Ferri (2019).

Como a imagem 7 nos mostra, a análise descritiva é responsável por avaliar como estão os dados agora e o que aconteceu no passado. A diagnóstica nos diz, ainda com base nos dados passados, porque o evento aconteceu de certa maneira, trazendo o diagnóstico. A preditiva (que quer dizer previsão), prevê futuros possíveis com base nos dados coletados no passado e aplicando modelos estatísticos. Enquanto a prescritiva prescreve quais as melhores decisões que devem ser tomadas de acordo com cada cenário, por exemplo, qual a melhor rota a ser feita por caminhões para reduzir os custos de uma empresa de Logística.

1.5. O que faz um Engenheiro de Dados?

Falamos dos conceitos principais: dados, informação, conhecimento, inteligência, Big Data e os tipos de análises, agora vamos entender o papel dos Engenheiros de Dados em todo esse “ecossistema".

A Engenharia de Dados é a área responsável por planejar, criar, manter e evoluir toda a estrutura de dados de uma organização. Pedro Tebaldi diz ainda que "Engenheiro de Dados é um tipo especializado de Engenheiro de Software que possibilita outros a responderem questões sobre grandes massas de dados, com restrições específicas de latência e tempo".

Este profissional deve principalmente:

- Planejar e desenhar o modelo dos dados utilizando as técnicas de modelagem.
- Coletar e entender a origem dos dados que podem ser úteis para o que precisa ser analisado e coletá-los interna ou externamente, através de consulta a banco de dados de diferentes formatos, consumo de API, processamento de textos ou dados estruturados, enfim, onde houver dado o engenheiro precisa trabalhar para os tornar disponíveis aos próximos níveis.
- Planejar e desenvolver o esquema de dados através da modelagem.
- Decidir a melhor estratégia para transformar e processar o dado: envolve a infraestrutura capaz de processar os dados em grandes volumes, em diversos formatos, em tempo hábil;
- Criar soluções de Data Warehouse. Vamos falar mais sobre isso nos próximos capítulos.
- Armazenamento: decidir onde armazenar o dado, bancos de dados, DWs, Data Lakers, repositórios diversos.
- Disponibilização/Acesso: envolve uma estrutura voltada para segurança cujo usuário esteja habilitado a acessar os resultados finais das análises.

Capítulo 2. Modelagem

Antes de falarmos sobre modelagem é importante entendermos o termo “abstração”. Pois os modelos nada mais são que as abstrações dos elementos do mundo real. Abstração de dados é a técnica de focar nos elementos principais dos dados, ou seja, no significado, nas características e como se relacionam entre si, qual a sua função no negócio, quais processos do negócio estes suportam e menos em como serão armazenados, como podem ser retornados à aplicação, qual o tipo do dado e outras informações internas. A modelagem de dados é exatamente a representação dos dados abstraídos em diferentes níveis, sendo o Modelo conceitual o de maior abstração e o Físico de menor, por já se preocupar mais no formato do dado em sua camada final.

2.1. Modelagem Conceitual

É o modelo de mais alto nível de abstração, ou seja, que está mais próximo da realidade dos usuários. É construído a partir dos resultados extraídos na fase de levantamento de requisitos. Levantamento de requisito é quando os projetistas de banco de dados entrevistam os usuários para entenderem como realizam suas atividades, como se relacionam com os processos e sistemas vigentes e documentam os resultados. Nesta fase não há qualquer menção às tecnologias ou ferramentas, mas apenas o entendimento dos agentes que executam as ações, suas características e como se relacionam.

O modelo conceitual mais conhecido é o MER, que nasceu em 1970 como uma grande revolução no mundo de armazenamento e manipulação de dados. Essa arquitetura foi primeiramente descrita por Edgar F. Codd, um pesquisador da IBM, em seu artigo “A Relational Model of Data for Large Shared Data Banks”. Lá ele defendeu “que nos modelos vigentes na época, Rede e Hierárquico, as aplicações que os utilizassem deveriam conhecer profundamente as estruturas internas destes bancos

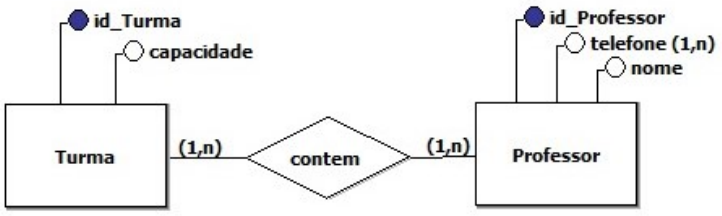
para navegar em busca dos dados. E ele propunha a criação de um modelo que permitisse maior abstração, em que as aplicações não precisassem conhecer o detalhamento do armazenamento, o que resultaria em melhor portabilidade. E foi então que ele descreveu o Modelo Relacional.

E o diagrama resultante do modelo conceitual MER é o DER (Diagrama de Entidade e Relacionamento), que nada mais é que a representação visual do que foi descrito e definido no MER. Geralmente são usados para comunicação com as camadas não técnicas, pois são de fácil entendimento e focam no funcionamento do negócio que se pretende armazenar e analisar.

Pode assumir diversas notações, mas possuem sempre os mesmos elementos: Entidades, Atributos e relacionamentos.

Exemplo do DER:

Figura 8 – Exemplo de notação de DER.



Entidade: Navathe explica que Entidade é o objeto básico do modelo ER, é algo no mundo real com uma existência independente. Pode ser abstrata, concreta ou associativa:

- Concreta: existe fisicamente no mundo real (carro, casa, pessoa, funcionário).

- Abstrata: não existe fisicamente, mas sim conceitualmente (empresa, pedido, disciplina).

- Fortes: as entidades fortes são aquelas cuja existência não depende de outra entidade. Já possuem sentido completo por si mesmas (funcionário, produto, carro).

- Fracas: entidades que só existem quando estão se relacionando com outra, por exemplo, itens do pedido só existem se tivermos pedido. O mesmo se dá para dependente em um sistema de RH, que só existe quando há um funcionário a qual se relacione.

- Associativa: entidade que existe como resultado do relacionamento "muitos para muitos", que será abordado em detalhe no tópico cardinalidade. Mas para explicar resumidamente, é quando em um relacionamento temos muitas ocorrências de uma entidade para muitas ocorrências da outra.

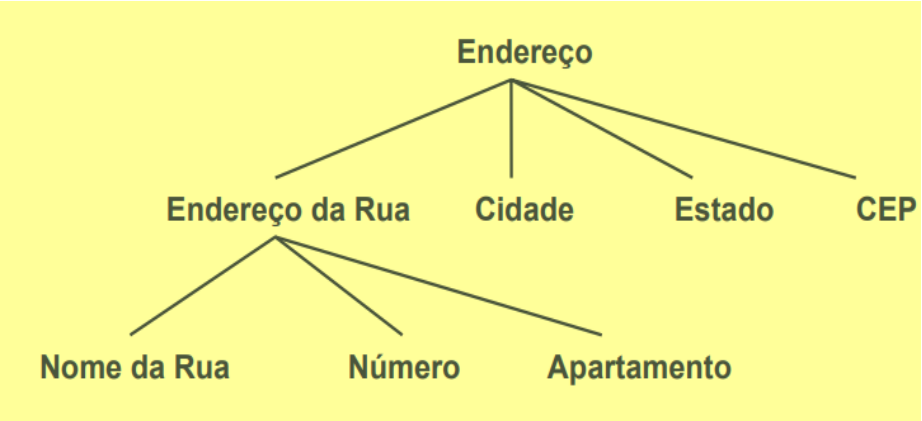
- Exemplo: um produto para estar em vários pedidos diferentes e um pedido pode ter vários produtos. Sendo um relacionamento muitos para muitos, como chamados, não ligamos as duas entidades diretamente e sim criamos uma tabela no meio para representar esse relacionamento, que seria Itens_Pedido, com a chave de Produto e Pedido, além de outros atributos que se deseje armazenar sobre os itens.

Atributo: ainda segundo Navathe (2011), atributos são características, propriedades específicas que descrevem as entidades. Eles podem ser simples, compostos, monovalorados, multivalorados, armazenados ou derivados:

- Simples: atributos atômicos, que não podem ser divididos em partes menores. Exemplo: Idade, CPF...

- Composto: atributos que podem ser divididos em subpartes menores, que representam atributos mais básicos com significados independentes. Exemplo: Endereço de um funcionário que poderia ser quebrado em vários atributos subpartes – Rua, número, Cidade, País.

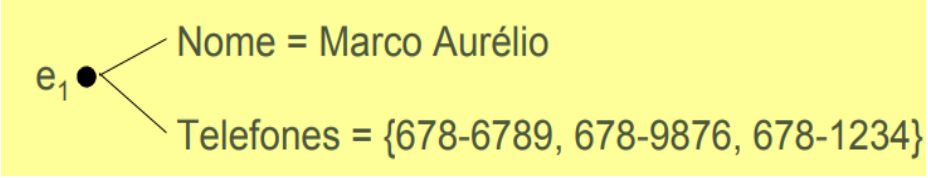
Figura 9 – Exemplo de atributo Composto.



- Monovalorado: a maioria dos atributos possui apenas um valor para uma entidade, estes são os monovalorados. Exemplo: idade, CPF, altura e outros. Uma pessoa só pode ter uma idade e um CPF.

- Multivalorado: já o atributo multivalorado é aquele que pode ter mais de uma ocorrência para uma mesma entidade. Por exemplo, o atributo Endereço pode aparecer como comercial e residencial, ou uma pessoa pode ter mais de um telefone ou ainda formação acadêmica. Neste caso, podemos definir quantidade mínima e máxima para essas ocorrências. Por exemplo: quero que alguém tenha no mínimo 1 telefone e no máximo 3). Não confundam composto com multivalorado. O composto é um atributo que pode ser quebrado em vários e multivalorado, um mesmo atributo mais de uma vez na entidade, telefone, formação acadêmica e outros.

Figura 10 – Exemplo de atributo Multivalorado.



- Armazenado: atributos com valores fixos persistidos em banco.

- Derivado: são atributos cujos valores devem ser obtidos após algum processamento utilizando informações obtidas do próprio banco de dados: Exemplo: idade = Data_Atual - Data_Nascimento ou Número de empregados de um determinado departamento, calculado a partir de uma contagem de ocorrências de empregado em relação ao departamento. No geral, a responsabilidade de trabalhar com dados derivados é da aplicação, sendo persistido apenas os atributos armazenados (explicado no tópico anterior).

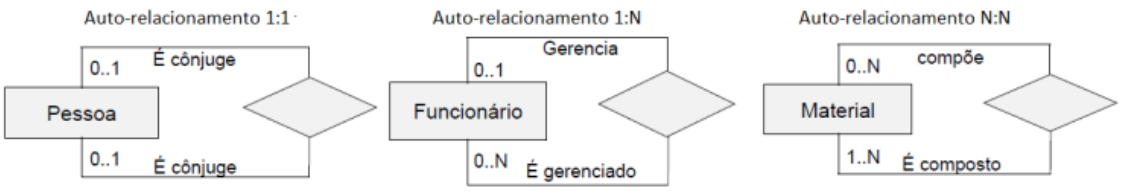
- Atributo chave: é aquele que representa unicamente uma ocorrência da entidade. Por exemplo: cod_cliente, CPF, CRM. Garante unicidade.

Relacionamento: um relacionamento é uma associação entre uma ou mais entidades. Exemplo: Funcionário trabalha em Departamento; Pedido possui Produtos. Estes relacionamentos podem ser classificados quanto ao grau:

- Binário/Ternário: é definido pela quantidade de entidades envolvidas no relacionamento. No unário temos apenas uma entidade, um auto relacionamento (explicaremos melhor a seguir), binário são duas entidades, ternário três e assim por diante. Exemplo: Empregado e Departamento se relacionando é um binário; Fornecedor fornece Produto para Projeto, temos três entidades envolvidas, portanto um ternário.

- Auto relacionamento, recursivo ou unário: antes de entendermos o auto relacionamento é importante explicar que entidades possuem papéis, ou uma "função" no relacionamento. Exemplo: funcionário **trabalha** em Departamento, Departamento **possui** Funcionário. Mas existem alguns casos em que a mesma entidade assume diferentes papéis, se tornando assim um relacionamento recursivo ou auto relacionamento, exemplo: Funcionário gerencia Funcionário:

Figura 11 – Auto relacionamento.

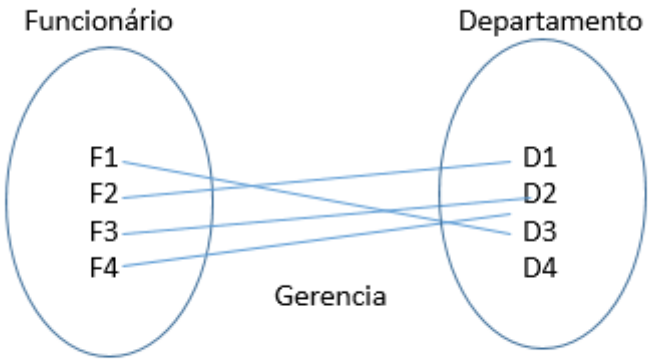


Os relacionamentos também podem ser definidos pela razão de cardinalidade.

A razão de cardinalidade especifica a quantidade de instâncias/ocorrências de relacionamentos em que uma entidade pode participar (1:1, 1:N, N:N).

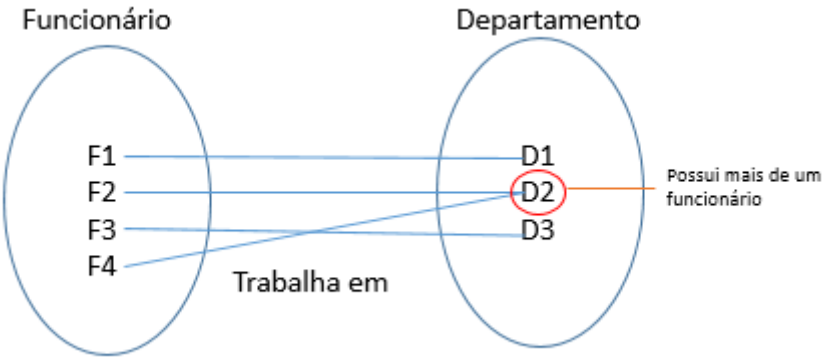
1:1 (um para um): **um** empregado gerencia **um** departamento e um departamento pode ser gerenciado por apenas **um** funcionário:

Figura 12 – Relacionamento 1:1.



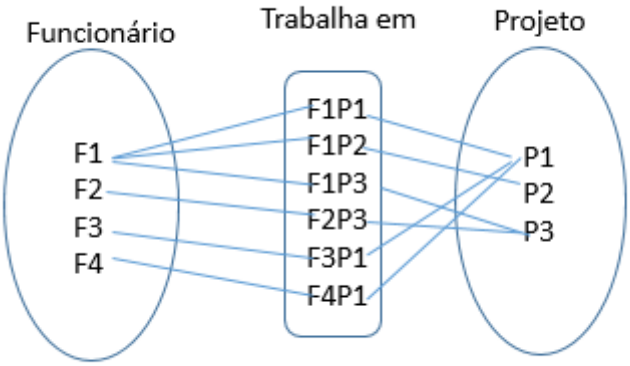
1:N ou N:1 (Um para muitos ou Muitos para um) – **Um** empregado trabalha em **um** departamento e **um** departamento pode ter **vários** (N) empregados:

Figura 13 – Relacionamento N:1 ou 1:N.



N:N (muitos para muitos): Empregado pode trabalhar em **vários** (N) projetos e um projeto pode possuir **vários** (N) empregados. Uma observação é que neste caso, para evitar redundância de dados, cria-se uma entidade associativa, por exemplo, Trabalha_Em, que receberá as chaves de Empregado e Projeto.

Figura 14 – Relacionamento N:N.



É importante salientar que a cardinalidade pode ser expressa com valores específicos e não apenas como N. Por exemplo, em determinado cenário a regra de negócio pode definir que uma ocorrência da entidade funcionário pode ter no máximo 3 projetos e não N (sem limitação), e assim teríamos (N:3).

Além da cardinalidade também especificamos a obrigatoriedade, ou seja, se uma ocorrência de uma entidade é obrigatória ou não. Exemplo: Se departamento precisa ter pelo menos um funcionário ao ser cadastrado este é obrigatório no relacionamento e representamos como 1 no DER. E caso não seja obrigatório representamos a obrigatoriedade como 0.

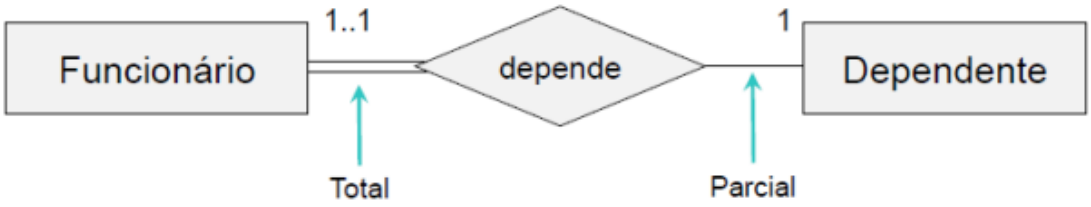
Então a notação do relacionamento no modelo ER fica sendo um parêntese, com o primeiro valor à esquerda representando a obrigatoriedade (0 para não – 1 para sim) e após a vírgula temos a quantidade de ocorrências possíveis no relacionamento (1,N,3,4,5 etc).

Figura 15 – Cardinalidade / Obrigatoriedade.

Outro conceito interessante é a ideia de participação no relacionamento. Ele especifica se a existência de uma entidade depende de ela estar relacionada com outra entidade através de um relacionamento ou não. Assim, a participação pode ser:

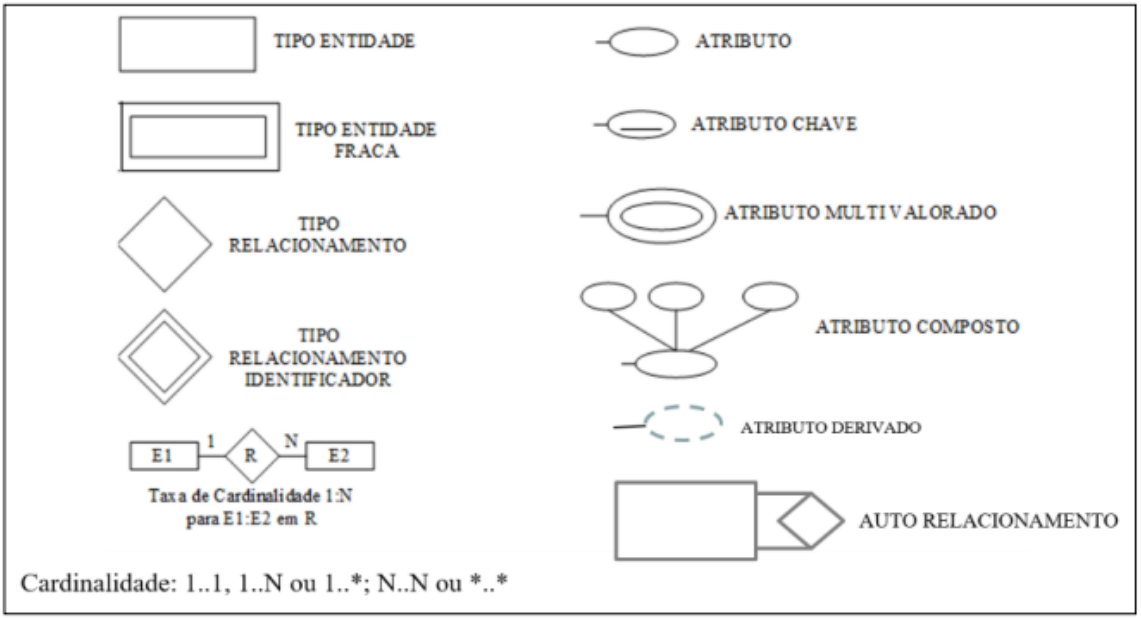
- Total: a entidade existe somente quando ela se relaciona com outra entidade.
- Parcial: a existência de uma entidade independe do fato dela estar relacionada com outra.

Figura 16 – Participação da entidade (Total/Parcial).



Como explicado, o MER é graficamente representado pelo DER (Diagrama de Entidade e Relacionamento) e existem diversas notações ou maneiras de se representar, sendo a mais conhecida a proposta por Peter Chen:

Figura 17 – Notação DER – Peter Chen.



2.2. Modelo Lógico

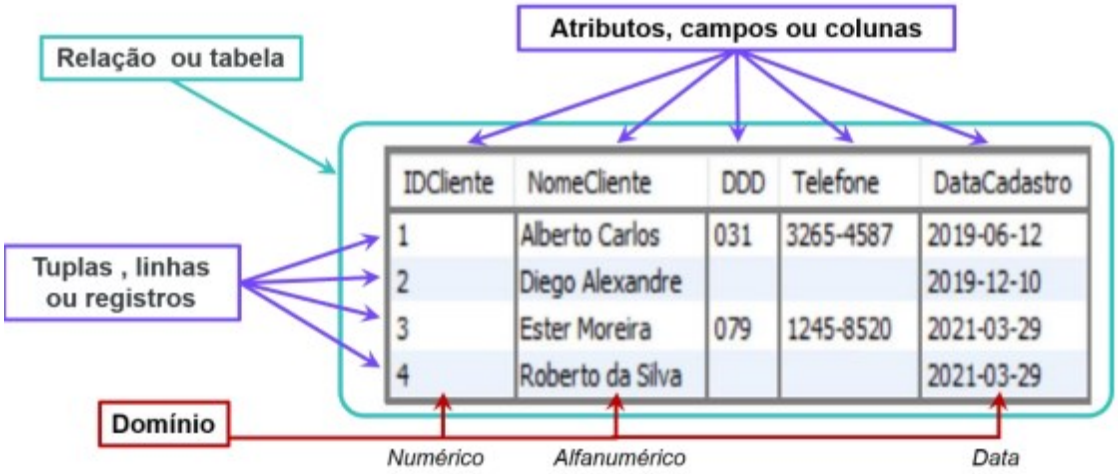
O modelo lógico é um aprofundamento do modelo conceitual. Neste modelo ainda não se escolheu a tecnologia, mas sim o paradigma: se será relacional, orientado a objeto, hierárquico, de rede, o que vai nos dizer sobre como os dados se organizarão e irão interagir entre si.

Modelo relacional

Quando o paradigma é relacional, os modelos lógicos resultantes são chamados de modelos relacionais e este é um dos mais importantes modelos lógicos. Portanto, com base no MER (artefato do modelo conceitual já explicado) avançamos mais um nível, descrevendo os dados em maiores detalhes, em preparação para a próxima etapa, a física.

Elsmari e Navathe (2011) explicam em seu livro “Sistemas de Banco de dados”, que o Modelo Relacional representa o banco de dados como uma coleção de relações, que no modelo conceitual chamamos de entidade e no lógico/físico, são as tabelas. Essas tabelas possuem atributos, ou colunas, que lhe conferem características e cada ocorrência dessa tabela são as linhas, tuplas ou registros. Um atributo possui ainda um domínio vinculado a ele, ou seja, um conjunto de valores atômicos que o atributo pode assumir.

Figura 18 – Tabela com suas colunas e linhas.



E como o nome já diz, no modelo relacional essas tabelas se relacionam através das chaves. As chaves são os atributos identificadores, ou seja, aqueles que identificam unicamente uma linha (chave primária e chave única), ou atributos que promovem o relacionamento entre tabelas (chave estrangeira).

Figura 19 – PK, UK e FK.

Explicando melhor esses conceitos:

Chave Primária (Do inglês Primary Key – PK): recupera uma única ocorrência do conjunto de dados, deve ser única e não nula. Força a integridade na tabela pois não permite inserir linhas com a chave duplicada, portanto é chamada de restrição (do inglês Constraint). Podem ser simples, formadas por uma única coluna, ou compostas, quando possuem várias colunas para garantir unicidade da tupla. Outra característica que define a chave primária é se é natural, com significado no mundo real (CPF, CNPJ, CRM) ou apenas uma chave aleatória sequencial criada pelo banco, a que chamamos de Surrogate key ou chave "burra". Uma tabela só permite uma PK e é preciso escolher a coluna chave com muito cuidado, pois alterá-la depois geralmente é bem difícil e causa impacto no uso da tabela durante a mudança. A PK de uma tabela se torna a FK de outra. Veremos o conceito de FK a seguir.

Chave única (do Inglês Unique Key – UK): as restrições UK também identificam uma tupla unicamente em uma tabela. Uma tabela pode ter mais de uma chave única ao contrário da chave primária. As restrições de chave única podem aceitar apenas um valor NULL para a coluna. Também podem ser referenciadas pela chave estrangeira de outra tabela, não somente as PKs, embora seja mais comum pela PK. Ou seja, em resumo, as chaves únicas podem ser usadas quando se deseja impor restrições exclusivas em uma coluna ou grupo de colunas que não são chave primária. Exemplo: Cod_Cliente já é a PK da tabela Cliente, mas quero garantir que o campo CPF não se repita na tabela. Dessa maneira posso criar uma chave única, UK, pelo campo CPF.

Chave estrangeira (do Inglês Foreing key - FK): chave estrangeira é quando para duas tabelas que se relacionam, o atributo (coluna) de uma das tabelas for chave primária ou chave única da outra tabela. Em outras palavras, é a coluna que promove o relacionamento entre duas tabelas. E vale ressaltar que uma FK pode ser nula se a relação não for obrigatória.

Exemplo: em um cenário em que temos duas tabelas que se relacionam, *Funcionário* e *Departamento*, onde um funcionário pode trabalhar em um departamento e um Departamento pode ter vários funcionários (1:N), a chave da tabela Departamento, Cod_Departamento, irá para a de Funcionário como uma nova coluna para que tenhamos esse relacionamento.

Figura 20 – Exemplo de FK.

Ao criarmos a tabela Funcionário adicionamos uma nova coluna Cod_Departamento e dizemos que esta faz referência à coluna Cod_Departamento da tabela Departamento. No capítulo de SQL veremos exemplos práticos.

Restrições ou Constraints: são campos ou regras criadas para garantir integridade dos dados de uma tabela. A língua portuguesa define integridade como a qualidade do "que não sofreu alteração, que não foi quebrado, que está ileso, incorruptível, correto e honesto". As chaves são um tipo de constraint mas temos outras que definem se o dado pode ou não ser nulo, se possui valor padrão caso nada seja informado na inserção, dentre outros. Vamos conhecer esses tipos a seguir:

Integridade de Chave: define os valores que não podem ser nulos nem duplicados. PK.

Integridade Referencial: define que os valores que aparecem como FK em uma tabela (no nosso exemplo Cod_Departamento) devem obrigatoriamente existir na tabela de Departamento.

Integridade de domínio: define os dados que determinada coluna pode receber, seja o tipo: texto ou número, ou opções rígidas disponíveis, estado civil: casado ou solteiro, ou sexo: feminino, masculino ou outros.

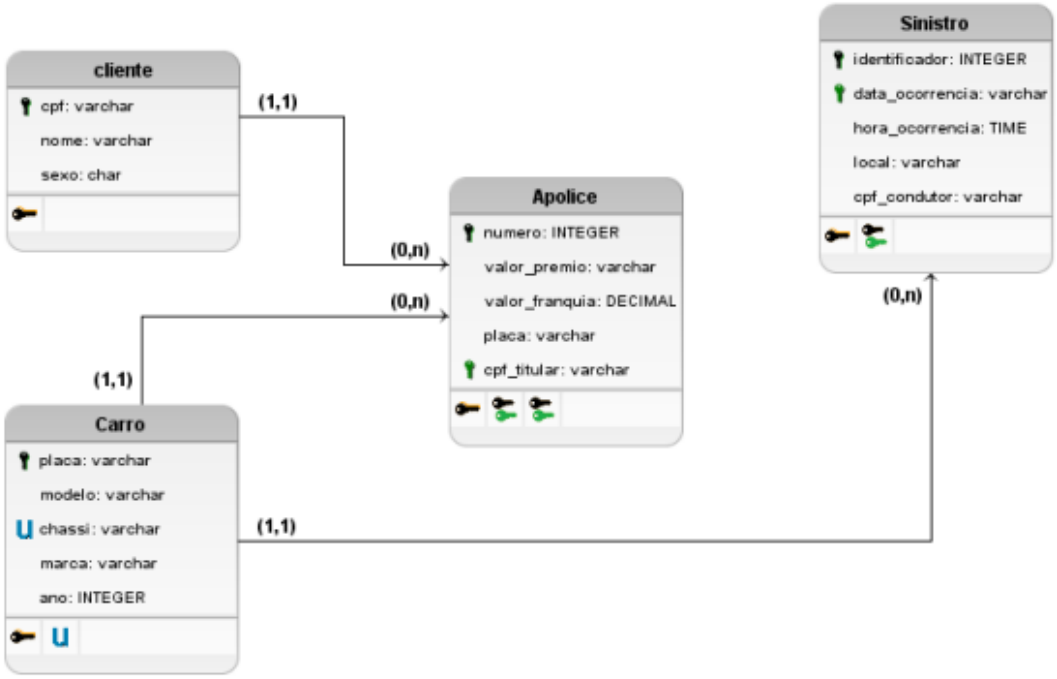
Integridade de Unicidade: define se os valores daquela coluna podem repetir. PK e UK.

Integridade de valor padrão: define se uma coluna possui valor padrão se nada for informado durante a inserção.

Integridade semântica: restrições criadas pelo usuário para limitar de acordo com a regra de negócio quais valores possíveis e aceitos para o campo. Por exemplo: valor de salário não pode ser menor que salário mínimo.

Integridade de vazio: neste caso, define se o valor pode ser nulo ou vazio. Vale ressaltar que atribuir valor vazio ou 0 a um campo não é o mesmo que nulo.

Figura 21 – Exemplo de modelo relacional.



2.3. Bancos Relacionais e SGBDs

Estudamos os conceitos de Dados e Modelagem de Dados, agora é o momento de entendermos onde esses dados podem ser armazenados. Uma das opções, e mais utilizada, são os bancos de dados. Mas o que é um banco de dados? Navathe (2011) explica que é uma coleção de dados inter-relacionados, representando informações sobre um domínio específico, ou seja, sempre que for possível agrupar dados que se relacionam e tratam de um mesmo assunto, posso dizer que tenho um banco de dados. Existem diversos tipos de bancos de dados, os relacionais, que implementam o paradigma relacional, armazenam dados estruturados e que utilizam a linguagem SQL (Structure Query Language) para manipular dados, e os não relacionais, que trabalham com dados não estruturados e são chamados de NoSQL ou Not only SQL, por utilizarem outras linguagens para manipular os dados.

Já o SGBD (Sistema Gerenciador de Banco de Dados) é responsável por gerenciar múltiplos bancos de dados, acessos simultâneos, abstrair o acesso a estes dados e permitir o controle de permissão. Basicamente podemos dizer que o sistema gerenciador de banco de dados:

Permite isolar o usuário ou aplicação da estrutura interna em que os dados são armazenados, ou seja, promove abstração, o que permite que estes interajam com o dado em mais alto nível.

Permite o acesso e manipulação dos dados: inserir, selecionar ou retornar, atualizar e apagar.

Permite o controle de permissões para que somente usuários/aplicações autorizadas tenham acesso. A maioria dos SGBDs segue a regra que "por padrão o usuário não possui permissão" e a mesma precisa ser explicitamente concedida por outro usuário com permissão.

Utiliza algoritmos e mecanismos para definir a melhor maneira de trabalhar com os dados, para garantir performance. Obviamente é necessário configurá-lo e gerenciá-lo devidamente, mas suas funcionalidades nativas garantem performance na maioria dos casos.

Como exemplos de Sistemas de Gerenciamento de Banco de Dados relacionais no mercado podemos citar: SQL Server, Oracle e Mysql. Para garantir consistência e bom funcionamento de seus dados, esses SGBDs relacionais precisam implementar um conceito chamado de ACID. É a base do universo do banco de dados relacional e precisamos discuti-lo adequadamente. De acordo com a Microsoft (2018), ACID é:

Atomicidade: ou a transação termina completamente ou será revertida. Em um grupo de comandos executados em um objeto, se um deles falhar, todo o processo será revertido para manter integridade.

Consistência: o mecanismo precisa manter o banco de dados consistente. Se algo der errado, ele trabalhará para voltar o banco de dados no último status consistente. E todos os usuários precisam obter as mesmas e corretas informações, de acordo com o que foi salvo e manipulado.

Isolamento: como o ambiente de banco de dados é frequentemente usado por vários usuários ao mesmo tempo, os bancos de dados relacionais tratam as transações como uma unidade e as isolam para garantir que, se uma pessoa precisar atualizar um campo/página, outra não possa ler, ou vice-versa.

Durável: se as informações foram salvas, elas precisam ser mantidas corretamente.

Para manipular dados nos SGBDs relacionais, utilizamos a linguagem SQL, como já mencionado, Structure Query Language ou Linguagem de consulta estruturada.

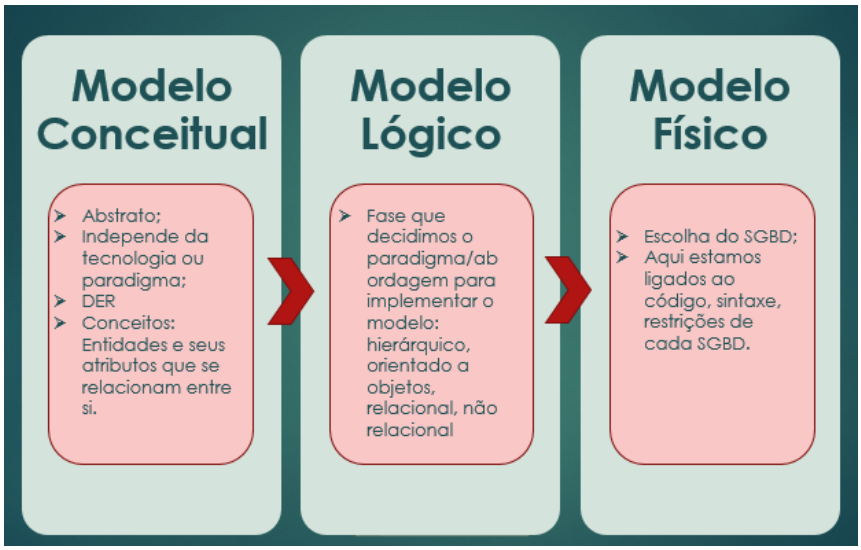
2.4. Modelo físico

Baseado no paradigma escolhido no modelo lógico, podendo ser relacional, hierárquico ou orientado a objetos, definimos a tecnologia que armazenará os dados, ou seja, o SGBD – Sistema Gerenciadores de Banco de Dados. E cada um dos SGBDs disponíveis no mercado terão algumas características próprias que o modelo físico deve considerar, mas independentemente de qual SGBD for escolhido, os padrões básicos serão consistentes com o paradigma em si. Em outras palavras, qualquer banco relacional do mercado implementará os conceitos relacionais (entidade, atributo, relacionamento, chaves...).

Sendo assim, baseado no resultado da modelagem lógica, esta é a fase de escolher o SGBD e criar o modelo de armazenamento utilizando alguma linguagem e respeitando as restrições impostas pela tecnologia escolhida.

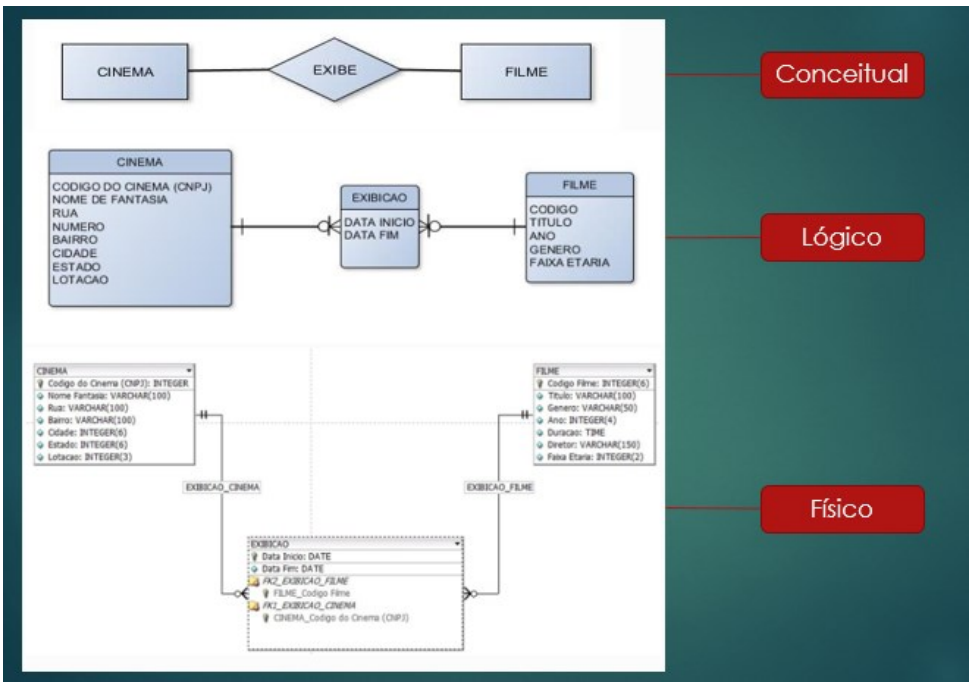
E o modelo mais utilizado e objeto de estudo deste módulo de fundamentos é o modelo relacional. Portanto, o SGBD escolhido será um que implemente tal arquitetura. Para citar algumas opções, temos: Microsoft SQL Server, Oracle, Mysql, Postgres, dentre outros.

Figura 22 – Comparação entre os modelos estudados.



E para entender as diferenças gráficas entre eles:

Figura 23 – Comparação gráfica entre os modelos estudados.



Capítulo 3. SQL

O SQL foi criado em meados da década de 70 como resultado do estudo de um laboratório da IBM em San Jose, Califórnia, seguindo o artigo de Codd mencionado no tópico anterior. O sucesso da linguagem foi tão grande que obrigou o American National Standards Institute (ANSI) a padronizar implementações em 1986, e o mesmo aconteceu com a International Organization for Standardization (ISO) em 1987. Portanto, os bancos relacionais compatíveis com SQL precisam utilizar os comandos básicos e principais no padrão ANSI/ISO, permitindo apenas algumas alterações em relação ao padrão. Isso facilita a criação de aplicações que consumam dados de diversos SGBDs diferentes ou até migrações, pois no geral pouco deverá ser alterado para se adequar às especificidades de cada um deles.

Através da linguagem SQL podemos realizar diversas tarefas nos SGBDs relacionais:

- Consultar dados no banco de dados – SELECT.
- Alterar dados – UPDATE.
- Inserir novos dados – INSERT.
- Apagar dados – DELETE/TRUNCATE.
- Alterar o esquema e a estrutura dos objetos – ALTER.
- Criar novos objetos (tabelas, usuários, procedures) – CREATE.
- Apagar objetos – DROP.
- Controlar permissão, concedendo ou negando – GRANT/REVOKE.

E para organizar estes comandos por seu objetivo e função, existem os grupos abaixo:

- DDL – Data Definition Language (Linguagem de Definição de Dados): Grupo de comandos responsáveis por criar ou apagar objetos (CREATE/DROP), alterar o esquema, estrutura ou tipo dos objetos de banco (ALTER). Habilitar e desabilitar objetos, como as triggers, para os SGBDs que possuem tal funcionalidade.

- DML – Data Manipulation Language (Linguagem de Manipulação de Dados): grupo de comandos usados para modificar os dados, as linhas de uma tabela – INSERT, UPDATE, DELETE.

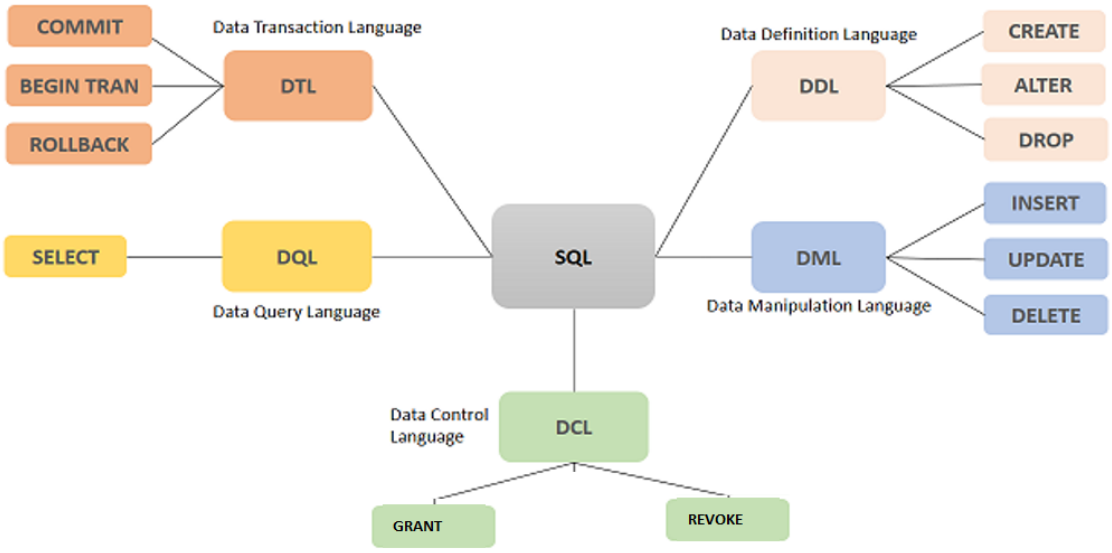
- Obs.: alguns autores adicionam TRUNCATE neste grupo, pois este comando apaga todas as linhas de uma tabela sem registrar logs que possibilitariam reverter a mudança. Como se trata de apagar linhas, a maioria considera DML, mas por também resetar os campos auto incrementais, outros autores colocam no grupo de DDL.

- DQL – Data Query Language (Linguagem de Consulta de Dados): é um conjunto de instruções usadas para consultar os dados (SELECT).

- DCL – Data Control Language (Linguagem de Controle de Dados): conjunto de comandos utilizados para controlar o acesso aos dados. Através deles podemos conceder (GRANT) ou negar (REVOKE). Alguns SGBDs também possuem o DENY.

- DTL – Data Transaction Language: Linguagem de Transação de Dados. São os comandos para controle de transação: BEGIN TRANSACTION, COMMIT E ROLLBACK.

Figura 23 – Comandos SQL.



Antes de iniciarmos o estudo sobre SQL em profundidade, vamos explicar o que são alguns outros objetos de Banco de Dados, além das tabelas, que os comandos SQL manipulam: VIEW, PROCEDURE, FUNCTION, TRIGGER.

- View (visões): a view pode ser definida como uma tabela virtual composta por linhas e colunas de dados vindos de tabelas relacionadas em uma query (um agrupamento de SELECT's, por exemplo). As linhas e colunas da view são geradas dinamicamente no momento em que é feita uma referência a ela.

- Index (índice): índice é uma referência associada a uma chave que é utilizada para fins de otimização, permitindo uma localização mais rápida de um registro quando efetuada uma consulta. Podemos fazer analogia ao índice de um livro, ou artigo, que nos ajudam a encontrar rapidamente os temas.

- Procedure (procedimento armazenado) ou Function (Função): o procedimento armazenado e a função são rotinas pré-compiladas, compostas por um

conjunto de comando SQL que são armazenados como objeto de banco de dados, que aceitam parâmetros e assim podem ser executadas repetidamente por meio de chamada ao procedimento. A diferença entre estas rotinas é que a função retorna um valor calculado dentro dela e a procedure não. A sintaxe de criação do procedimento armazenado e da função pode variar conforme o SGBD.

- Trigger ou Gatilho: uma Trigger é um procedimento armazenado no banco de dados, que é chamado automaticamente sempre que ocorre um evento especial no banco de dados. Este evento pode ser uma inclusão, alteração ou exclusão de dados em tabelas.

- Usuários: são aqueles que interagem com os bancos de dados dentro de um Sistema Gerenciador de Banco de Dados e para os quais concedemos ou retiramos permissões.

- Roles: uma Role ou papel é um agrupamento de permissões que pode ser concedida a usuários ou outras roles. Sua utilização ajuda a administrar as permissões de objetos no banco de dados e poupa o tempo que seria gasto com permissões e revogações individuais. A maioria dos SGBDs relacionais possuem algumas roles padrões para conceder permissão de leitura, alteração, administrador, mas também permite a criação de novas roles para atender às demandas do negócio.

- Privilégios: são os tipos de ações que um usuário pode realizar no banco de dados. No geral, os bancos de dados trabalham com o conceito de: “por padrão o usuário não tem permissão”.

3.1. DDL

Como explicado anteriormente a linguagem de definição de dados (DDL) é uma linguagem de computador usada para criar e modificar a estrutura de objetos de banco de dados relacional. Esses objetos incluem visões (views), esquemas, tabelas, índices, procedimentos armazenados (procedures), funções, gatilhos (triggers) etc.

É importante consultar a documentação do SGBD escolhido para obter a sintaxe específica. A seguir temos alguns exemplos genéricos:

Criar (Create):

CREATE DATABASE nome_do_banco_de_dados;

CREATE TABLE nome_da_tabela lista_de_opções_para_criação]

Exemplo padrão ANSI:

```
CREATE TABLE tabela (
  atrib1 tipo [(tamanho)] [NOT NULL | DEFAULT valor]
              [CHECK (condição)],
  atrib2 tipo [(tamanho)] [NOT NULL | DEFAULT valor]
              [CHECK (condição)],
    ...
  [CONSTRAINT nome da restrição] PRIMARY KEY (<atributos chave primária>),
  [CONSTRAINT nome da restrição] UNIQUE (< atributos chave candidata>),
  [CONSTRAINT nome da restrição] FOREIGN KEY (<atributos chave estrangeira>)
                REFERENCES tabelaRef [(<chave primária>)]
                    [ON DELETE CASCADE | SET NULL | SET DEFAULT]
                    [ON UPDATE CASCADE | SET NULL | SET DEFAULT],
   [CONSTRAINT nome da restrição] CHECK (condição)
);
```

CREATE VIEW nome_da_view AS commando_SELECT_que_vai_gerar _a view

CREATE PROCEDURE nome (lista opcional de parâmetros) AS;

CREATE FUNCTION nome (lista opcional de parâmetros) AS RETURNS;

Excluir (Drop)

```
DROP DATABASE nome_do_banco_de_dados;
DROP PROCEDURE nome_da_procedure;
DROP FUNCTION nome_da_procedure;
DROP VIEW nome_da_view;
DROP INDEX nome_da_tabela.nome_do_indice; (MSSQL)
DROP INDEX nome_do_indice (ORACLE);
ALTER TABLE nome_da_tabela DROP INDEX nome_do_indice (MYSQL)
```

Alterar (Alter):

```
ALTER qual_objeto_deseja_alterar (DATABASE, TABLE...) opção_de_alteração;
MYSQL e Oracle usam esse formato: MODIFY COLUMN nome_coluna tipo [opções de alteração]
```

3.2. DML

A linguagem de manipulação de dados (Data Manipulation Language, DML) é a sublinguagem do SQL, que permite aos usuários manipular dados, ou seja, incluir dados em uma tabela, alterar ou excluir.

É importante consultar a documentação do SGBD escolhido para obter a sintaxe específica. A seguir temos alguns exemplos genéricos:

Atualizar (Update):

É o comando DML usado para modificar dados existentes em uma tabela, sejam eles dados individuais ou grupos de dados. Não confunda com o ALTER que altera a estrutura, o esquema da tabela. O UPDATE pode conter a cláusula WHERE para definir quais linhas serão atualizadas.

A sintaxe básica do comando possui a seguinte estrutura:

```
UPDATE nome_tabela SET nome_atributo_1 = valor [{,nome_atributo_n = valor}] [WHERE condição];
```

Deletar (Delete):

O comando DELETE remove linhas de uma tabela. Ele também pode incluir a cláusula WHERE para filtrar o que será excluído. A exclusão ainda pode ser feita comparando as chaves de mais de uma tabela, através de JOINS, que serão explicados na sessão do DQL.

```
DELETE FROM nome_tabela [WHERE condição];
```

Inserir (Insert):

O comando INSERT em sua forma mais básica é usado para acrescentar uma linha a uma tabela. Para tal temos que informar o nome da tabela e a cadeia de valores que desejamos inserir em cada uma das colunas. Os valores devem ser informados na mesma ordem em que as colunas foram especificadas.

```
INSERT INTO nome_tabela[(lista_atributos)] VALUES(lista_valores_atributos) [,
(lista_valores_atributos)];
```

3.3 DQL

A Linguagem de Consulta de Dados ou Data Query Language (DQL) é a sublinguagem do SQL responsável por realizar consultas (leitura) aos dados de uma determinada tabela do banco de dados. O único comando deste grupo é o SELECT, que pode ser desde consultas simples com apenas uma tabela, até complexas, envolvendo várias tabelas, filtros, consultas aninhadas e outros elementos.

O SELECT consiste em uma projeção dos dados do banco, onde informamos quais colunas queremos trazer, todas ou algumas, se queremos adicionar agregação – Soma, Contagem, Média – de onde os dados serão projetados, de uma tabela ou de

várias, se queremos filtrar essa projeção, usando o comando WHERE, se queremos ordenar através do ORDER BY e assim por diante.

A sintaxe padrão é:

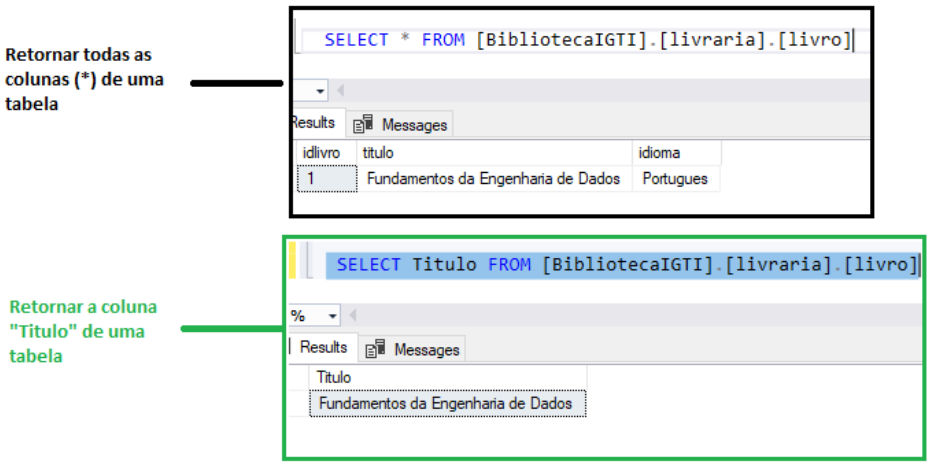
```
SELECT [predicado { * | tabela.* | [tabela.]campo1 [AS alias1] [, [tabela.]campo2
[AS alias2] [, ...]]}
FROM expressãotabela [, ...]
[WHERE... ]
[GROUP BY... ]
[HAVING... ]
[ORDER BY... ]
```

Todas as colunas versus colunas específicas:

Para consultar todos os campos de uma tabela utilize o * após o comando SELECT, caso queira especificar as colunas a serem retornadas, informe-as separadas por vírgula:

```
SELECT * FROM nome_tabela
SELECT nome_coluna1, nome_coluna2... FROM nome_tabela
```

Exemplo:



SELECT valores distintos:

Use o DISTINCT para informar que não quer retornar valores duplicados.

```
SELECT * FROM Livros
```

Results | Messages

Titulo	Idioma
Fundamentos da Engenharia de Dados	Portugues
Fundamentos da Engenharia de Dados	Portugues

A tabela Livros possui valores duplicados

Ao adicionarmos DISTINCT * retornamos as linhas que possuem valores diferentes para todas as colunas.

```
SELECT DISTINCT * FROM Livros
```

Results | Messages

Titulo	Idioma
Fundamentos da Engenharia de Dados	Portugues

Caso quiséssemos retornar valores DISTINCTS para a coluna Titulo bastaria ao invés do * adicionar SELECT DISTINCT Titulo FROM...

SELECT ordenados:

Para ordenar o resultado do SELECT adicione ORDER BY nome_coluna [n...] [ASC] ou [DESC], sendo ASC ascendente, do menor para o maior e DESC, descendente, do maior para o menor. ASC é o padrão para o ORDER BY caso nada seja informado.

Caso a cláusula ORDER BY não seja informado, os registros serão retornados na ordem em que foram inseridos.

```
CREATE TABLE TESTORDER (Numeros INT)
INSERT INTO TESTORDER VALUES (25), (47), (5),(1)
SELECT * FROM TESTORDER
```

Results | Messages

Numeros
25
47
5
1

Como não foi informada ordem de retorno no SELECT, os registros foram retornados na ordem que foram inseridos

Outro exemplo – Listar os nomes distintos das editoras por ordem alfabética:

```
SELECT DISTINCT nome_editora
FROM editora
ORDER BY nome_editora;

SELECT DISTINCT nome_editora
FROM editora
ORDER BY nome_editora ASC;
```

nome_editora
Casa dos Espiritos
Editora Lê
Manole
Moderna
Objetiva

Listar os nomes distintos das editoras por ordem alfabética decrescente:

```
SELECT DISTINCT nome_editora
FROM editora
ORDER BY nome_editora DESC;
```

nome_editora
Objetiva
Moderna
Manole
Editora Lê
Casa dos Espiritos

Operadores aritméticos, relacionais e condicional:

Os operadores aritméticos são utilizados para realizar operações matemáticas simples no SELECT, como soma, subtração, multiplicação e divisão.

+	Adição
-	Subtração
*	Multiplicação
/	Divisão

Já os relacionais ou condicionais são aqueles que nos permitem filtrar os resultados do SELECT, por exemplo, se são maiores ou menores que determinado valor, se não são nulos, além de definir se todas as condições precisam ser verdadeiras (AND) ou apenas algumas (OR).

Figura 24 – Operadores SQL.

Condições	
=	igual
<	menor
<=	menor ou igual
>	maior
>=	maior ou igual
<>	diferente
AND	todas as condições verdadeiras
OR	pelo menos uma condição verdadeira
NOT	não satisfaz a condição
BETWEEN ... AND	dentro de um intervalo
LIKE	pesquisar por padrão
IN	possíveis valores

Exemplo: considerando a tabela:

titulo_livro	preco
Pelas Ruas de Calcutá	36.1
Devoted - Devoção	27.2
Xô, Bactéria! Tire Suas Dúvidas Com Dr. Bactéria	32.7
P.s. - Eu Te Amo	23.5
O Que Esperar Quando Você Está Esperando	37.8
As Melhores Frases Em Veja	23.9
Bichos Monstruosos	24.9
Casas Mal Assombradas	27.9
Memórias Póstumas de Brás Cubas	22.5
Dom Casmurro	25.9
Dom Casmurro	35.9
Quincas Borba	35.9

Através do SELECT abaixo podemos somar, subtrair, multiplicar ou dividir a coluna Preço e projetar o resultado.

```
SELECT titulo_livro, preco, preco + 10 AS adição, preco - 10 AS subtração, preco * 0.1 AS multiplicação, preco / 10 AS divisão
FROM livro;
```

titulo_livro	preco	adição	subtração	multiplicação	divisão
Pelas Ruas de Calcutá	36.1	46.099998474121094	26.099998474121094	3.6099998474121096	3.6099998474121096
Devoted - Devoção	27.2	37.20000076293945	17.200000762939453	2.7200000762939456	2.720000076293945
Xô, Bactéria! Tire Suas Dúvidas Com Dr. Bactéria	32.7	42.70000076293945	22.700000762939453	3.2700000762939454	3.2700000762939454
P.s. - Eu Te Amo	23.5	33.5	13.5	2.35	2.35
O Que Esperar Quando Você Está Esperando	37.8	47.79999923706055	27.799999237060547	3.779999923706055	3.779999923706055
As Melhores Frases Em Veja	23.9	33.89999961853027	13.899999618530273	2.3899999618530274	2.3899999618530274
Bichos Monstruosos	24.9	34.89999961853027	14.899999618530273	2.4899999618530275	2.4899999618530275
Casas Mal Assombradas	27.9	37.89999961853027	17.899999618530273	2.7899999618530273	2.7899999618530273
Memórias Póstumas de Brás Cubas	22.5	32.5	12.5	2.25	2.25
Dom Casmurro	25.9	35.89999961853027	15.899999618530273	2.5899999618530276	2.589999961853027
Dom Casmurro	35.9	45.900001525878906	25.900001525878906	3.5900001525878906	3.5900001525878906
Quincas Borba	35.9	45.900001525878906	25.900001525878906	3.5900001525878906	3.5900001525878906

E usando os condicionais abaixo conseguimos filtrar valores maiores ou menores que 30.

```
SELECT titulo_livro, preco
FROM livro
WHERE preco > 30;
```

titulo_livro	preco
Pelas Ruas de Calcutá	36.1
Xô, Bactéria! Tire Suas Dúvidas Com Dr. Bactéria	32.7
O Que Esperar Quando Você Está Esperando	37.8
Dom Casmurro	35.9
Quincas Borba	35.9

```
SELECT titulo_livro, preco
FROM livro
WHERE preco < 30;
```

titulo_livro	preco
Devoted - Devoção	27.2
P.s. - Eu Te Amo	23.5
As Melhores Frases Em Veja	23.9
Bichos Monstruosos	24.9
Casas Mal Assombradas	27.9
Memórias Póstumas de Brás Cubas	22.5
Dom Casmurro	25.9

Filtros

Um usuário geralmente está procurando por informações específicas no Banco de Dados. Deste modo, há a utilização do comando WHERE (ONDE), juntamente com alguns comandos específicos para só retornar ou projetar os resultados desejados.

A sintaxe padrão é:

SELECT *coluna1, coluna2, …*
FROM *nome_tabela*
WHERE *condição*

Vamos a alguns exemplos para entender os filtros. Para tal considere a tabela abaixo:

Aluno				
matrícula	nome_aluno	endereço	disciplina	nota
111	Caio	Rua 1	Português	5
111	Caio	Rua 1	Física	9
222	Fernando	Rua 2	Matemática	10
333	João	Rua 3	Português	3
333	João	Rua 3	História	8
444	César	Rua 4	Física	8

Tabela Principal

Retornar todos os alunos que estudam Português:

```
SELECT *
FROM Aluno
WHERE disciplina = 'Português'
```

Aluno				
matrícula	nome_aluno	endereço	disciplina	nota
111	Caio	Rua 1	Português	5
333	João	Rua 3	Português	3

Comando WHERE

- Operadores AND / OR:

A cláusula WHERE pode ser utilizada com mais de uma condição. Para isso, podem ser empregados os comandos AND ou OR. O comando AND é utilizado para retornar valores em que todas as condições dispostas sejam verdadeiras, já no OR apenas uma delas precisa ser verdade para ser retornado.

Ainda utilizando a tabela Aluno, caso precisemos retornar todos os alunos que chamam Caio e estudam Português, utilizamos o primeiro exemplo, já para trazer quem chama Caio ou estuda Português, utilizamos o segundo comando:

```
SELECT *
FROM Aluno
WHERE nome_aluno='Caio' AND disciplina='Português'
```

Aluno				
matrícula	nome_aluno	endereço	disciplina	nota
111	Caio	Rua 1	Português	5

Comando AND

```
SELECT *
```

FROM *Aluno*

WHERE *nome_aluno='Caio'* OR *disciplina='Português'*

Aluno				
matrícula	nome_aluno	endereço	disciplina	nota
111	Caio	Rua 1	Português	5
111	Caio	Rua 1	Física	9
333	João	Rua 3	Português	3

Comando OR

- NOT: o NOT é utilizado quando se quer apenas aqueles registros que não satisfazem uma determinada condição. Por exemplo, é possível pesquisar todos os registros em que a nota não seja 8:

SELECT *

FROM *Aluno*

WHERE NOT *nota=8*

Aluno				
matrícula	nome_aluno	endereço	disciplina	nota
111	Caio	Rua 1	Português	5
111	Caio	Rua 1	Física	9
222	Fernando	Rua 2	Matemática	10
333	João	Rua 3	Português	3

Comando NOT

Vale ressaltar que o NOT pode ser substituído por outros operadores, o <>, por exemplo no SQL Server, para dizer é diferente de 8.

- Operador LIKE:

Este é um operador um pouco mais complexo. Ele é utilizado quando se quer encontrar registros específicos dentro de um campo texto. Por exemplo, seria possível encontrar apenas os alunos que possuem nome que comece com a letra C ou nomes que terminem com a letra O? Sim, seria possível através do comando LIKE.

O comando LIKE é utilizado juntamente com os caracteres de porcentagem (%) e do underline (_). A porcentagem (%) define que pode ser qualquer valor antes ou depois dela, por exemplo '%a', quer dizer que o texto pode ter qualquer valor antes de a com qualquer quantidade de caracteres. Já o Underline (_) é para o caso do usuário querer definir a quantidade de caractere. Exemplo: Texto que comece com A, com obrigatoriamente 5 caracteres, utiliza-se: 'A_ _ _ _'.

Outros exemplos:

Figura 25 – Exemplos do operador Like.

Comando	Retorna
LIKE 'A%'	Qualquer string que seja inciada com a letra A.
LIKE '%A'	Qualquer string que seja iniciada com a letra A.
LIKE '%A%'	Qualquer string que tenha a letra A em qualquer posição.
LIKE 'A_'	String que possua dois caracteres, sendo a primeira letra A e a segunda qualquer outra.
LIKE '_A'	String que possua dois caracteres, sendo a última a letra A e a primeira qualquer outra.
LIKE '_A_'	String que possua exatamente três caracteres, em que o segundo seja obrigatoriamente a letra A.
LIKE '_ _ _'	Qualquer string com exatamente três caracteres.
LIKE '%A_'	Qualquer string que tenha a letra A na penúltima posição e a última seja qualquer outro caractere, possuindo qualquer tamanho.
LIKE '_ _ _%'	Qualquer string com pelo menos três caracteres.

Sintaxe padrão:

```
SELECT coluna1, coluna2, ...
FROM nome_tabela
WHERE coluna LIKE padrão
```

O primeiro comando retornará todos os alunos que começam com C sem definir quantidade de caracteres. Já o segundo filtra os que começam com C, mas possuem 4 caracteres.

```
SELECT *
FROM Aluno
WHERE nome_aluno LIKE 'C%'
```

Aluno				
matrícula	nome_aluno	endereço	disciplina	nota
111	Caio	Rua 1	Português	5
111	Caio	Rua 1	Física	9
444	César	Rua 4	Física	8

Comando LIKE %

```
SELECT *
FROM Aluno
WHERE nome_aluno LIKE 'C_ _ _'
```

Aluno				
matrícula	nome_aluno	endereço	disciplina	nota
111	Caio	Rua 1	Português	5
111	Caio	Rua 1	Física	9

Comando LIKE _

- Operador BETWEEN: O BETWEEN … AND é utilizado para procurar dados que estão dentro de um determinado intervalo de valores. A sua estrutura é a seguinte:

```
SELECT coluna1, coluna2, …
FROM nome_tabela
WHERE coluna BETWEEN valor1 AND valor2
```

Este comando retorna todos os alunos que possuem nota entre 2 e 6:

```
SELECT *
FROM Aluno
WHERE nota BETWEEN 2 AND 6
```

Aluno				
matrícula	nome_aluno	endereço	disciplina	nota
111	Caio	Rua 1	Português	5
333	João	Rua 3	Português	3

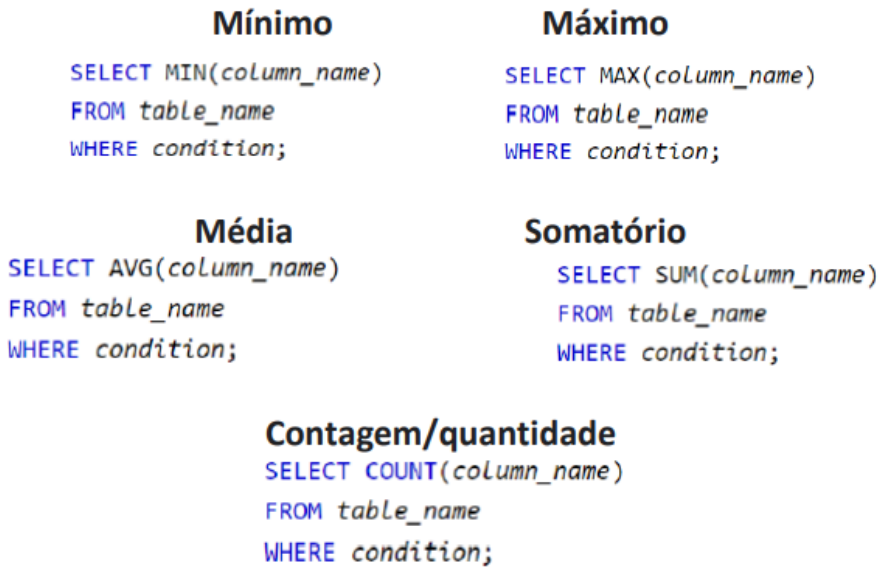
Comando BETWEEN

Agregadores

Uma função de agregação executa um cálculo em um conjunto de valores e retorna um único valor. As principais funções de agregação são:

- COUNT: o total de linhas.
- SUM: a soma total.
- MAX: o valor máximo.
- MIN: o valor mínimo.
- AVG: a média aritmética.

Figura 26 – Exemplos de agregação.



Agrupamentos e o comando HAVING:

Podemos usar as cláusulas GROUP BY junto aos comandos de agregação mencionados acima para obter os totais, agrupando por uma ou mais colunas. Já o Having nos permite filtrar os resultados das agregações. Sintaxe:

```
SELECT column_name(s)
FROM table_name
WHERE condition
GROUP BY column_name(s);
```

```
SELECT column_name(s)
FROM table_name
WHERE condition
GROUP BY column_name(s)
HAVING condition;
```

Considerando a tabela abaixo:

titulo_livro	preco
Pelas Ruas de Calcutá	36.1
Devoted - Devoção	27.2
Xô, Bactéria! Tire Suas Dúvidas Com Dr. Bactéria	32.7
P.s. - Eu Te Amo	23.5
O Que Esperar Quando Você Está Esperando	37.8
As Melhores Frases Em Veja	23.9
Bichos Monstruosos	24.9
Casas Mal Assombradas	27.9
Memórias Póstumas de Brás Cubas	22.5
Dom Casmurro	25.9
Dom Casmurro	35.9
Quincas Borba	35.9

AVG (preco) = média de preço;
(+) group by tilulo_livro = média de preço por título do livro;
(+) having avg(preco) < 30 = média de preço por título do livro, mas só retorne aqueles com média menor que 30

```
SELECT titulo_livro, avg(preco)
FROM livro
GROUP BY titulo_livro
HAVING avg(preco) < 30;
```

titulo_livro	avg(preco)
Devoted - Devoção	27.200000762939453
P.s. - Eu Te Amo	23.5
As Melhores Frases Em Veja	23.899999618530273
Bichos Monstruosos	24.899999618530273
Casas Mal Assombradas	27.899999618530273
Memórias Póstumas de Brás Cubas	22.5

Consultas aninhadas ou subconsulta

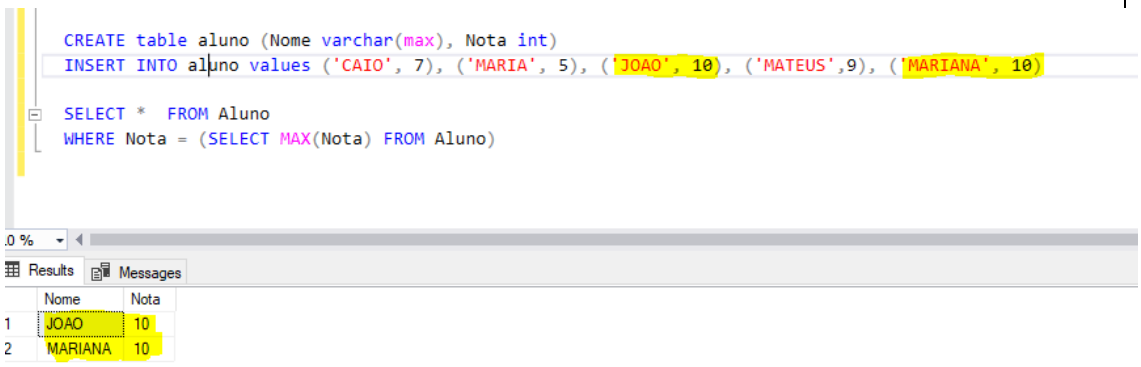
Uma subconsulta é uma consulta que está aninhada dentro de outra instrução SELECT.

Exemplo:

```
SELECT *
FROM Aluno
WHERE Nota = (SELECT MAX(Nota) FROM Aluno).
```

Neste exemplo está sendo retornado todos os alunos que tem nota igual a maior nota da tabela. Se o sub comando “SELECT MAX(Nota) From Aluno” retornar que a maior nota é 7, então todos os alunos da tabela que possuem nota 7 serão retornados.

Existem outras maneiras de retornar essa mesma informação, mas nesse exemplo usamos a subconsulta, ou uma consulta “dentro da outra”.



Joins

No modelo relacional, como o nome já diz, podemos ter várias tabelas se relacionando entre si através das chaves. E o JOIN ou junção é a operação capaz de retornar os registros destes relacionamentos. Portanto, para manipular dados de várias tabelas relacionadas por suas Foreign Keys, utilizamos os comandos descritos em detalhe a seguir, a depender do tipo de operação que desejamos. Todas as demais

cláusulas (WHERE, Agrupamento, Agregação, Condicional...) podem ser combinadas aos JOINS.

Sintaxe padrão:

```
SELECT lista de colunas
FROM tabela1 [INNER | LEFT | RIGHT | FULL ] JOIN tabela2 ON tabela1.coluna = tabela2.coluna;
```

Onde selecionamos os registros de uma tabela JOIN com outra, comparando as chaves FK de uma com a PK/chave única de outra na cláusula ON.

JOINS são a aplicação da teoria de conjuntos, conceito da matemática, em que analisamos os registros como participantes ou não de outro grupo de registros. Exemplo: considerando o grupo de números A= {1,2,5,25,85,100} e B= {5,100}, quais registros estão nos dois? 5 e 100, quais estão somente no primeiro? 1, 2, 25 e 85. Quais estão somente no segundo 5, 100, e como ficaria a junção total de ambos? {1,2,5,25,85,100}. E é exatamente o que fazemos com as tabelas através dos JOINS no universo do banco relacional. Então vamos entender os tipos de JOINS:

- (INNER) JOIN: a cláusula **INNER JOIN** compara cada linha da tabela A com as linhas da tabela B para encontrar todos os pares de linhas que satisfazem a condição de junção (ON). Se a condição de junção for avaliada como TRUE, os valores da coluna das linhas correspondentes das tabelas A e B serão combinados em uma nova linha e incluídos no conjunto de resultados.

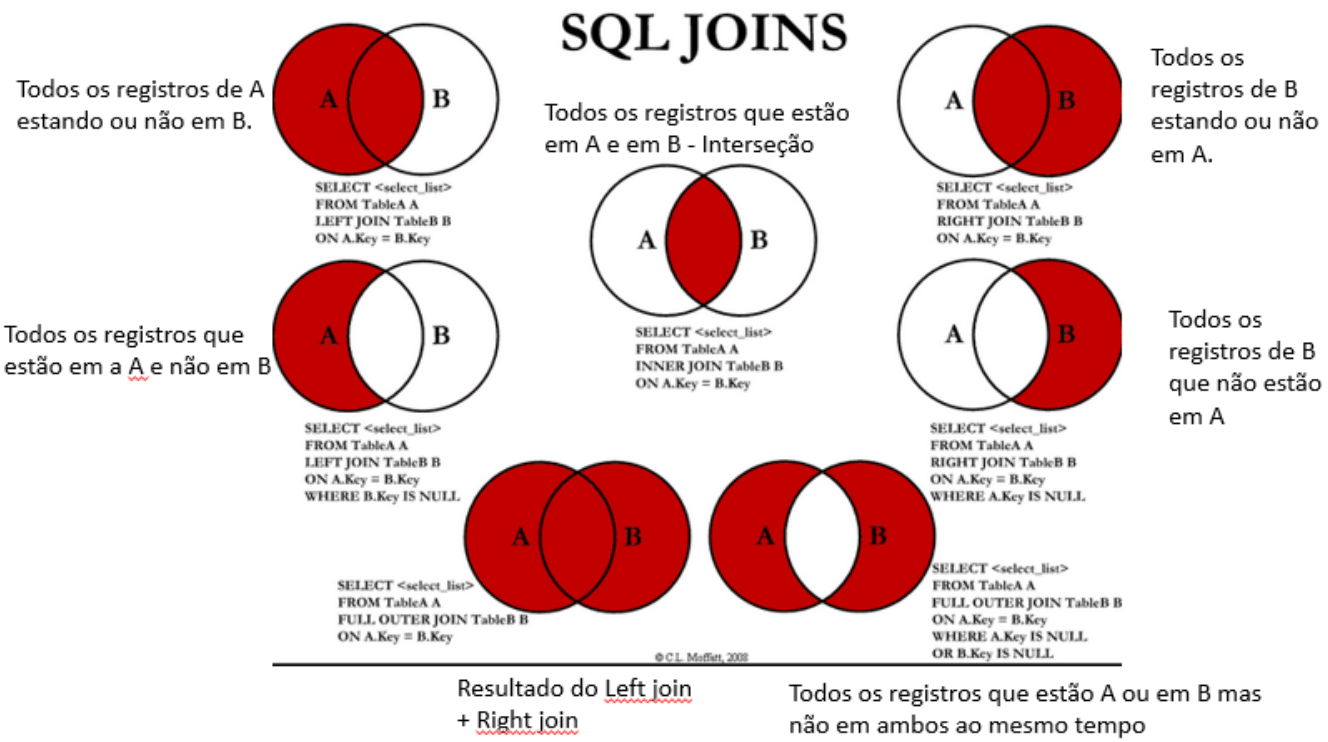
- LEFT (OUTER) JOIN: para cada linha da tabela A, a consulta a compara com todas as linhas da tabela B. Se um par de linhas fazer com que a condição de junção seja avaliado como TRUE, os valores dessas linhas serão combinados para formar uma nova linha que será incluída no conjunto de resultados. Se uma linha da tabela da “esquerda” A não tiver nenhuma linha correspondente da tabela “direita” B, a consulta irá combinar os valores da coluna da linha da

tabela “esquerda” A com NULL para cada valor da coluna da tabela da “direita” B que não satisfaça a condição de junção (FALSE). Em resumo, a cláusula LEFT JOIN retorna todas as linhas da tabela “esquerda” A e as linhas correspondentes ou valores NULL da tabela “direita” B.

- RIGHT (OUTER) JOIN: combina dados de duas ou mais tabelas através de suas chaves, selecionando dados da tabela “direita” B e comparando às linhas da tabela “esquerda” A. O RIGHT JOIN retorna um conjunto de resultados que inclui todas as linhas da tabela “direita” B, com ou sem linhas correspondentes na tabela “esquerda” A. Se uma linha na tabela direita B não tiver nenhuma linha correspondente da tabela “esquerda” A, a coluna da tabela “esquerda” A no conjunto de resultados será nula igualmente ao que acontece no LEFT JOIN.

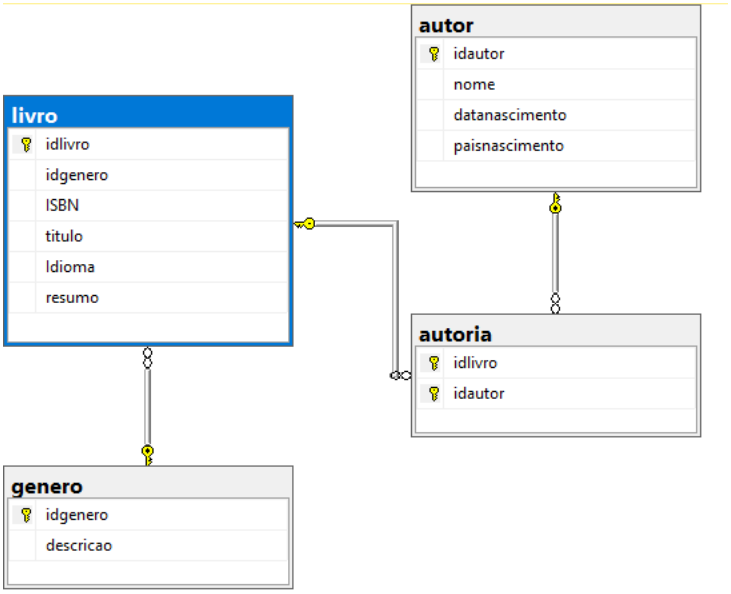
- FULL (OUTER) JOIN: compara todas as linhas de uma tabela com a outra, retornando as linhas que possuem correspondência ou não. A cláusula FULL JOIN retorna todas as linhas das tabelas unidas, correspondidas ou não, ou seja, você pode dizer que a FULL JOIN combina as funções da LEFT JOIN e da RIGHT JOIN.

Figura 27 – JOINS.



Exemplo:

Considerando as tabelas, Autor, Livro, Gênero, Autoria no esquema abaixo:



Onde Livro se relaciona com Gênero através da FK idgenero e Autoria se relaciona com Autor e Livro pelas FKs Idlivro, idautor.

Autoria

idlivro	idautor
3	3

Livro

idlivro	idgenero	ISBN	titulo	Idioma	resumo
3	1	4554654654	A menina que roubava livros	Portugues	Teste
4	3	658565465	Fundamentos em Engenharia de dados	Portugues	Teste
6	NULL	545454545	O segredo	Portugues	Test

Gênero

idgenero	descricao
1	Romance
2	Suspense
3	Técnicos

Autor

idautor	nome	datanascimento	paisnascimento
3	Maria	1985-01-01	Brasil
4	José	1990-02-05	Brasil

JOIN – Retornar todas os livros e seus respectivos gêneros:

```
SELECT * FROM Livro JOIN GENERO
ON Livro.idgenero = GENERO.idgenero
```

Results | Messages

idlivro	idgenero	ISBN	titulo	Idioma	resumo	idgenero	descricao
3	1	4554654654	A menina que roubava livros	Portugues	Teste	1	Romance
4	3	658565465	Fundamentos em Engenharia de dados	Portugues	Teste	3	Técnicos

FK

PK de Gênero

LEFT JOIN – Retornar todos os gêneros tendo livros referenciados ou não:

A esquerda do Left

A direita do LEFT

```
SELECT * FROM genero LEFT JOIN Livro
ON genero.idgenero = livro.idgenero
```

Results | Messages

idgenero	descricao	idlivro	idgenero	ISBN	titulo	Idioma	resumo
1	Romance	3	1	4554654654	A menina que roubava livros	Portugues	Teste
2	Suspense	NULL	NULL	NULL	NULL	NULL	NULL
3	Técnicos	4	3	658565465	Fundamentos em Engenharia de dados	Portugues	Teste

Como para o gênero Suspense (a esquerda do LEFT) eu não tenho Livros cadastrados, os campos de Livro aparecem em nulo

LEFT JOIN com WHERE IS NULL – Retornar todos os Gêneros que não possuem Livros cadastrados.

Podemos pegar a consulta anterior e filtrar onde o campo chave de Livro é nulo no relacionamento. Isso trará as não correspondências entre as duas tabelas. Suspense, portanto, é um gênero que não possui livros cadastrados.

```
SELECT * FROM genero LEFT JOIN Livro
ON genero.idgenero = livro.idgenero
WHERE Livro.idlivro IS NULL
```

esults | Messages

idgenero	descricao	idlivro	idgenero	ISBN	titulo	Idioma	resumo
2	Suspense	NULL	NULL	NULL	NULL	NULL	NULL

RIGHT JOIN: retornar todos os gêneros tendo livros referenciados ou não:

Para este exemplo apenas invertemos a ordem das tabelas Livro e Gênero em relação ao LEFT. Isso porque o RIGHT retorna todas as informações da tabela da direita (Gênero), tendo Livro (Esquerda) ou não.

```
SELECT * FROM Livro RIGHT JOIN genero
ON genero.idgenero = livro.idgenero
```

esults | Messages

idlivro	idgenero	ISBN	titulo	Idioma	resumo	idgenero	descricao
3	1	4554654654	A menina que roubava livros	Portugues	Teste	1	Romance
NULL	NULL	NULL	NULL	NULL	NULL	2	Suspense
4	3	658565465	Fundamentos em Engenharia de dados	Portugues	Teste	3	Técnicos

FULL OUTER JOIN: retorna todos os Livros tendo ou não gênero e todos os gêneros tendo ou não Livros cadastrados:

idlivro	idgenero	ISBN	titulo	Idioma	resumo	idgenero	descricao
3	1	4554654654	A menina que roubava livros	Portugues	Teste	1	Romance
4	3	658565465	Fundamentos em Engenharia de dados	Portugues	Teste	3	Técnicos
6	NULL	545454545	O segredo	Portugues	Test	NULL	NULL
NULL	NULL	NULL	NULL	NULL	NULL	2	Suspense

3.4. DCL

A Linguagem de Controle de Dados, ou do inglês Data Control Language (DCL), é uma sub-linguagem do SQL, usada para controlar o acesso aos dados e objetos em um banco de dados. Os comandos DCL permitem conceder e revogar permissões ou privilégios de acesso a objetos do banco de dados para usuários e papéis (roles):

Usuários: aqueles que terão acesso ao banco de dados para realizar as tarefas. Pode ser uma aplicação, usuário final ou grupo de usuários;

Permissões: permissão é o que o usuário ou role podem executar no banco de dados: podemos conceder permissão de leitura, escrita, de administrador e outros.

Nível de banco, server, objeto: as permissões podem ser concedidas a nível de SGBD, para executar atividades de administrador de banco de dados, a nível de banco de dados ou até a nível de objetos.

Roles: uma role ou papel é um agrupamento de permissões que podem ser concedidas a usuários ou outras roles e serve para facilitar a administração e manutenção das permissões. Os bancos geralmente já possuem suas roles padrões, mas nos permitem criar outras para atender melhor nossas necessidades.

Conceder permissão (Grant):

```
GRANT ON TO [WITH GRANT OPTION] [GRANTED BY grantor];
```

Sendo Grantor o usuário que concedeu permissão. WITH GRANT OPTION é a cláusula que diz se você deseja que esse usuário conceda permissões a outros. O padrão é não.

```
GRANT ALL PRIVILEGES ON exemplo TO 'dba';
GRANT SELECT ON exemple.* TO 'read';
GRANT INSERT, UPDATE, DELETE ON exemple.* TO 'write';
GRANT SELECT, INSERT, UPDATE, DELETE ON exemple.* TO 'read_write';

GRANT 'read_write' TO 'developer';
GRANT CREATE, DROP, REFERENCES, ALTER, EXECUTE, CREATE VIEW, TRIGGER ON *.* TO developer;
```

Remover permissão (Revole).

REVOKE[GRANT OPTION FOR] ON objeto FROM [RESTRICT|CASCADE]

```
REVOKE DELETE FROM 'read_write';

REVOKE 'read' FROM 'usuario3'@'localhost';

REVOKE INSERT, UPDATE, DELETE ON exemple.* FROM 'usuario1';
```

É importante salientar que para todos os comandos SQL cada banco possui seus próprios padrões de sintaxe, resguardados aqueles definidos como obrigatórios para bancos relacionais pelo padrão ANSI. Então é importante consultar a documentação do SGBD escolhido.

Pratique em: https://www.w3schools.com/sql – Navegando entre as páginas com cada um dos comandos.

Capítulo 4. Modelagem dimensional

Modelagem dimensional (DM) é uma modelagem focada no armazenamento de dados em Data Warehouse. Data Warehouse por sua vez é um repositório central de informações que são usados para análises do negócio e tomada de decisão. Os dados salvos no DW fluem de diversas fontes após serem transformados, agrupados, organizados e sumarizados. Os bancos de dados tradicionais são transacionais, ou utilizados para suporte à produção da empresa, o seu dia a dia, e são chamados de OLTP (Processamento de transação On-line), já o Data Warehouse serve às aplicações de apoio à decisão (OLAP) e possuem as características a seguir:

- Implementa visão multidimensional, ou seja, com a visão de um fato de negócio em diferentes dimensões. Como em um cubo.

- Orientado por assunto: é focado em um assunto do negócio em particular, não no conjunto de operações. Exemplo: ao invés de focar nos aspectos que compõe a transação de pedido, o faz no fato Venda e suas métricas.

- Integrado: ainda que o DW contenha dados de diferentes fontes, em que uma mesma informação pode ser salva em formatos diversos, por possuir a fase de transformação e sanitização, eles chegam integrados e consistentes ao DW.

- Não volátil: em regra geral o DW é estável, ou seja, os seus dados não são modificados como nos ambientes transacionais (exceto para eventuais correções), mas apenas os dados novos são carregados e acessados para leitura. Isso possibilita análise histórica confiável;

E para que fique clara a diferença, vamos pensar no exemplo de uma empresa de vendas de produtos pela internet. Se no ambiente OLTP para suportar e registrar as transações temos as tabelas Produto, Vendedor, Pedido, Itens de Pedido, no ambiente

DW as tabelas de Pedido e Itens pedido se tornariam somente Venda, que é o “fato” que desejo realizar análises, o que chamo de tabela fato, e as tabelas ligadas a este fato e que caracterizam a venda, como produto e vendedor, são chamadas de dimensões.

No diagrama abaixo temos a comparação entre OLTP em OLAP em mais detalhes:

Figura 28 – Diagrama comparativo OLTP e OLAP.

	OLAP	OLTP
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.
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 dimensional. 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	O armazenamento é feito em estruturas de *Data Warehouse* com otimização no desempenho em grandes volumes de dados.	O armazenamento é feito em sistemas convencionais de banco de dados através dos sistemas de informações da organização.
Abrangência	É utilizado pelos gestores e analistas para a tomada de decisão.	É utilizado por técnicos e analistas e engloba vários usuários da organização.
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).	A 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.
Tipos de permissões nos dados	É permitido apenas a inserção e leitura. Sendo que para o usuário está apenas disponível a leitura.	Podem ser feito leitura, inserção, modificação e exclusão dos dados.

Os elementos de um DW são:

Fatos: os fatos são os assuntos/ocorrências de um negócio que serão objeto de análise, ou seja, acontecimentos do negócio que são merecedores de análise e controle na organização. A tabela de fatos é a principal tabela de um modelo dimensional, onde as métricas estão armazenadas. É composta por uma chave primária (combinação única das chaves de todas as dimensões a que se relaciona) e pelas métricas de interesse para o negócio. A tabela de fatos deve representar uma unidade do processo do negócio e não devem misturar assuntos diferentes numa mesma tabela.

Dimensão: as Dimensões são os descritores dos dados oriundos da Fato. São entidades do negócio que apresentam alguma influência sobre o fato em análise. Caracterizam o fato. É a Dimensão que permite a visualização das informações por diversos aspectos e perspectivas. Costuma-se dizer que as dimensões representam "pelo que se deseja filtrar ou agrupar os fatos". Exemplo: considerando-se a fato Venda, desejo visualizar por produto, por dia, por país, portanto temos as dimensões produto, tempo e país.

Métricas: as métricas são responsáveis por mensurar, monitorar e gerir as estratégias de uma empresa. São compostas por valores numéricos, atômicos e armazenadas nas tabelas Fato de um Data Warehouse. Dentro de um planejamento estratégico elas têm como objetivo fornecer informações o nível gerencial de uma organização.

Exemplos: Valor total Faturado, Número de Funcionários Ativos, Valor da Inadimplência, dentre outros.

As métricas podem ser de três principais tipos:

- Aditiva: as métricas aditivas são aquelas que podem ser sumarizadas independentemente das dimensões utilizadas. Este tipo de métrica pode ser utilizada sem quase nenhuma restrição ou limitação e são flexíveis o suficiente

para gerar informações em quaisquer perspectivas. Por exemplo, métricas como quantidade e valores de determinados itens podem ser, em geral, sumarizados por data (dia, mês ou ano), local, clientes, entre outras dimensões, sem perder a consistência da informação.

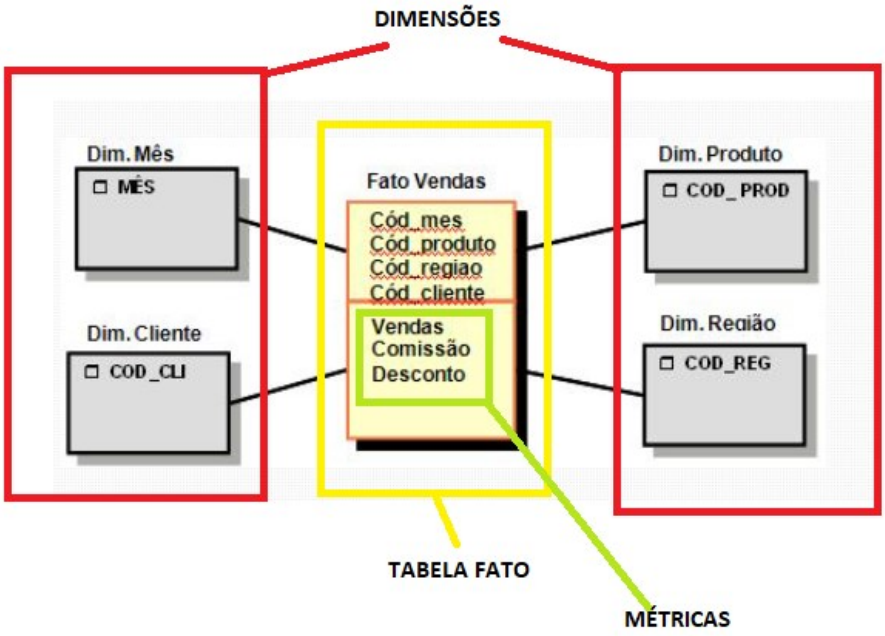
- Semi-aditiva: as métricas semi-aditivas são aquelas que podem ser sumarizadas em alguns casos, mas não em todos. Isso porque a depender da situação ela pode perder sentido para a análise. Nesse caso, a sumarização só fará sentido com algumas dimensões específicas. Por exemplo a métrica de saldo bancário. Essa métrica é um valor que reflete a situação atual da conta, que pode ter o saldo credor ou devedor. Faria sentido somar os saldos de todos os dias de um mês para uma determinada conta bancária? Claro que não. Pois se um dia o saldo for de -1000 e no dia seguinte ter os mesmos -1000, a soma irá me devolver um saldo negativo de -2000, o que não é verdade. Mas há casos onde a métrica semi-aditiva adquire característica de aditiva. Se por acaso somar os saldos de várias contas bancárias em um determinado dia, poderemos ver o saldo geral, o que tem total sentido e utilidade para uma instituição bancária.

- As métricas não-aditivas são aquelas que não podem ser sumarizadas ao longo das dimensões. Essas métricas não podem ter agregações pois perdem a veracidade do valor. Percentuais são exemplos de valores armazenados nas métricas que não permitem sumarizações. Por exemplo, não faz sentido algum somar o percentual de vendas de um item “A” que teve 50% de saída, com um item “B” que teve 60%. A soma resultaria em um valor agregado de 110%. O que isso nos diz? Nada!

KPIs ou Indicadores chave de desempenho são criados a partir das métricas. São eles que indicam o desempenho da empresa em suas diversas áreas e indicam o

possível caminho para alcançar-se o objetivo final. Exemplos de indicadores: Indicadores de qualidade e Indicadores de capacidade.

Figura 29 – Fatos, Dimensões e Métricas.



Os benefícios de um Data Warehouse incluem o seguinte:

- Tomada de decisão adequada.
- Dados consolidados de várias fontes.
- Análise de dados históricos.
- Qualidade, consistência e precisão de dados.
- Separação do processamento analítico dos bancos de dados transacionais, o que melhora o desempenho dos dois sistemas.

4.1. Granularidade

Outro conceito importante é a granularidade. Granularidade dos dados é a concepção que determina o armazenamento e tratamento dos dados em diferentes unidades ou níveis de detalhamento. Portanto, diz respeito à escala pelo qual se quer analisar os dados. Ela é determinada para cada tabela Fato, já que normalmente as Fatos possuem informações e granularidades distintas.

É importante entender o relacionamento existente entre o detalhamento e a granularidade. Quando falamos de menor granularidade (menor sumarização), ou granularidade fina, significa maior detalhamento dos dados. Maior granularidade (maior sumarização), ou granularidade grossa, significa menor detalhamento. Portanto, as grandezas granularidade e detalhamento são inversamente proporcionais.

Figura 30 – Granularidade x Detalhamento.

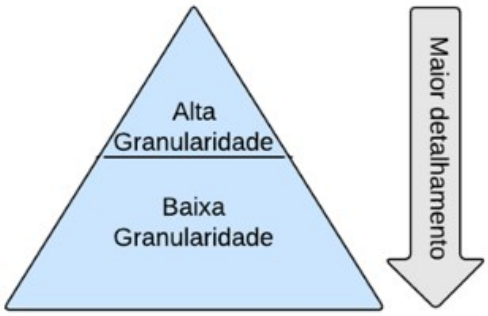
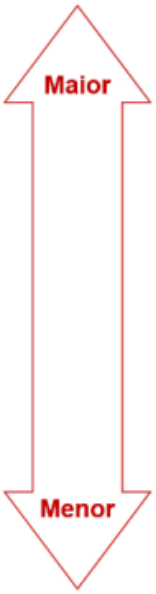


Figura 31 – Granularidade x Detalhamento – Dados.

Ocorrência	2017	2018	2019
Assaltos à mão armada	123	109	158
Furtos em residências	75	90	101
Furtos de veículos	243	250	332
Assassinatos	89	77	167
Estupros	2	24	69

Ocorrência	1º sem 2017	2º sem 2017	1º sem 2018	2º sem 2018	1º sem 2019	2º sem 2019
Assaltos à mão armada	50	73	45	64	70	88
Furtos em residências	40	35	60	30	71	30
Furtos de veículos	121	123	120	130	165	167
Assassinatos	40	49	37	40	67	100
Estupros	2	0	14	10	30	39



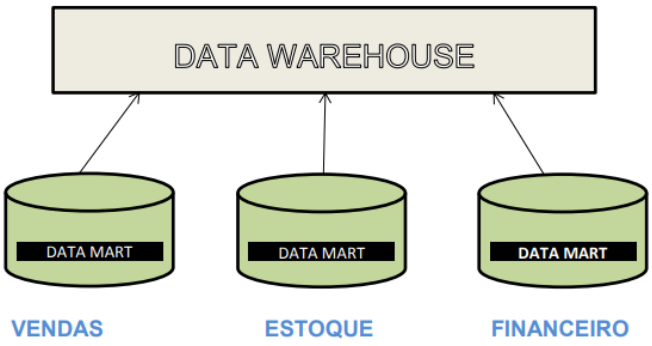
Imagine uma fato Vendas com sua dimensão de Tempo. A granularidade vai definir se os dados de vendas estarão disponíveis por ano, mês, minuto ou segundo. Então quanto menor a granularidade, mais detalhes teremos, neste exemplo, vendas por segundo.

É importante salientar que a granularidade impacta diretamente no volume de dados e na velocidade de processamento. Quanto maior o detalhamento, mais dados serão gerados e maior será o espaço consumido e os recursos computacionais para processar. E outro fator é, mudar a granularidade para grandes Fatos pode ser um grande problema, por isso a fase de modelagem, feita com qualidade e pensando nas demandas futuras, é tão importante.

4.2. Data Marts

Data Marts (mercado de dados) é uma forma simples de Data Warehouse com foco em um único assunto ou linha de negócios. Com um Data Mart as equipes podem acessar dados e obter insights mais rapidamente, pois não precisam gastar tempo pesquisando em um Data Warehouse mais complexo ou agregando manualmente dados de diferentes fontes. Alguns autores costumam dizer que um DW é um conjunto de Fatos, com suas tabelas fatos e dimensões, e Data Marts seria uma ou algumas Fatos e não toda a estrutura.

Figura 32 – Granularidade x Detalhamento – Dados.



4.3. Star schema vs Snowflake

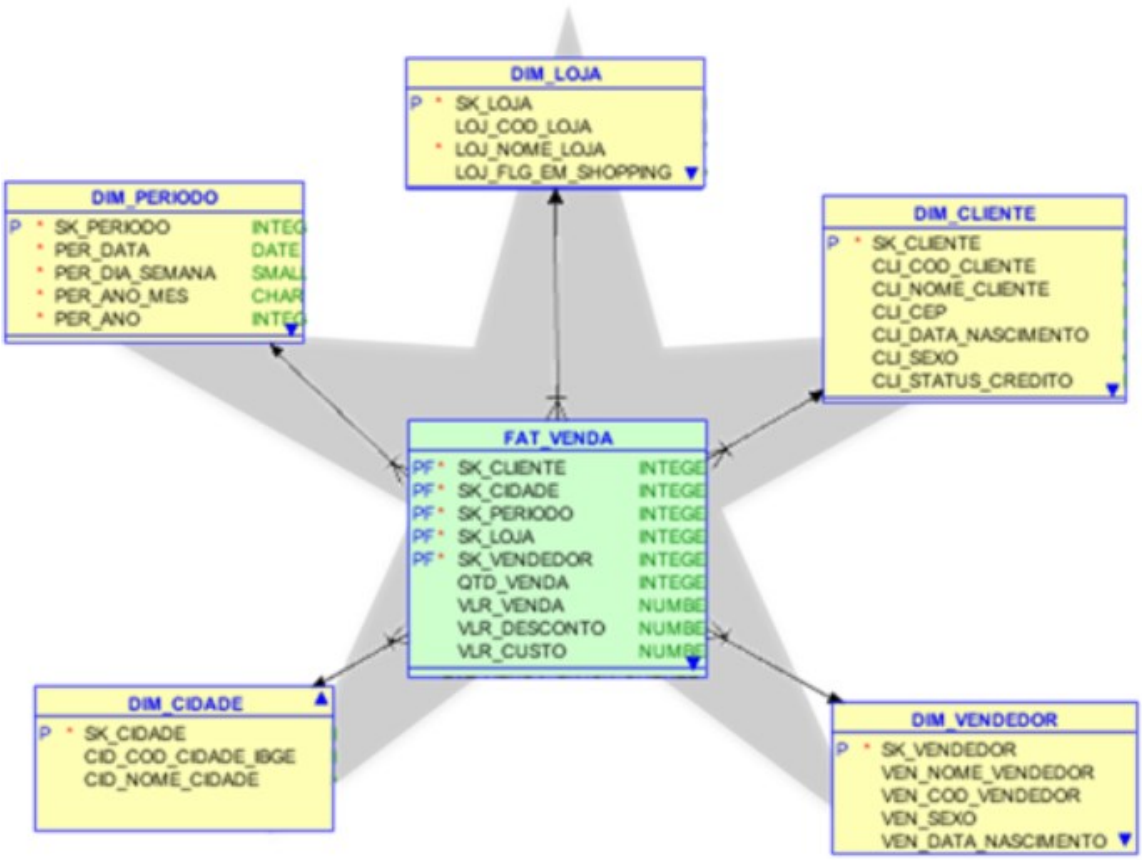
Escolher a melhor configuração para o DW é importante e deve ser feito com cuidado levando em consideração as necessidades do negócio. Existem basicamente duas maneiras de desenhar o seu DW, segundo o modelo Star Schema (Esquema Estrela) ou Snowflake (Esquema floco de nede).

No Star Schema as dimensões não são normalizadas, ou seja, todas as chaves das dimensões estão diretamente ligadas na fato, não havendo hierarquia entre dimensões. Isso acarreta maior duplicidade de dados no DW, pois para cada registro novo de uma dimensão não normalizada, todas as colunas devem ser repetidas e não só

a chave, mas confere maior performance. Já no modelo Snowflake, as dimensões estão normalizadas e possuem relacionamento com outras tabelas, assim joins adicionais precisam ser feitos entre fato e dimensões para se resgatar a informação completa.

Antes de escolher qual utilizar, vale esclarecer que no esquema estrela a navegação pelas tabelas é mais simples e rápida, pois cada join adicionado degrada a performance, porém utiliza mais espaço, repetindo as mesmas descrições ao longo de toda a estrutura. O Esquema Snowflake reduz a redundância, mas aumenta a complexidade e o tempo de resposta, pois possui mais tabelas envolvidas e mais joins são necessários para retornar toda informação. Portanto é preciso entender o que é a prioridade para o cliente/cenário, evitar redundância ou otimizar a performance do DW.

Figura 33 – Star Schema.



Perceba na Figura 31 que as dimensões não são normalizadas e a fato se relaciona somente com um nível de dimensões e a dimensão somente com a fato. Essa característica gera dados repetidos para agrupamentos, como na dimensão de CIDADE:

DIM_CIDADE			
SK_CIDADE	CID_COD_CIDADE_IBGE	CID_NOME_CIDADE	CID_UF
1	3509502	Campinas	SP
2	3513801	Diadema	SP
3	3550308	São Paulo	SP
4	4202404	Blumenau	SC
5	4205407	Florianópolis	SC
6	4208203	Itajaí	SC
7	4301602	Bagé	RS
8	4313409	Novo Hamburgo	RS
9	4314902	Porto Alegre	RS

Dados duplicados para a coluna CID_UF na dimensão Cidade

E como seria no Snowflake?

Figura 32 – Snowflake.

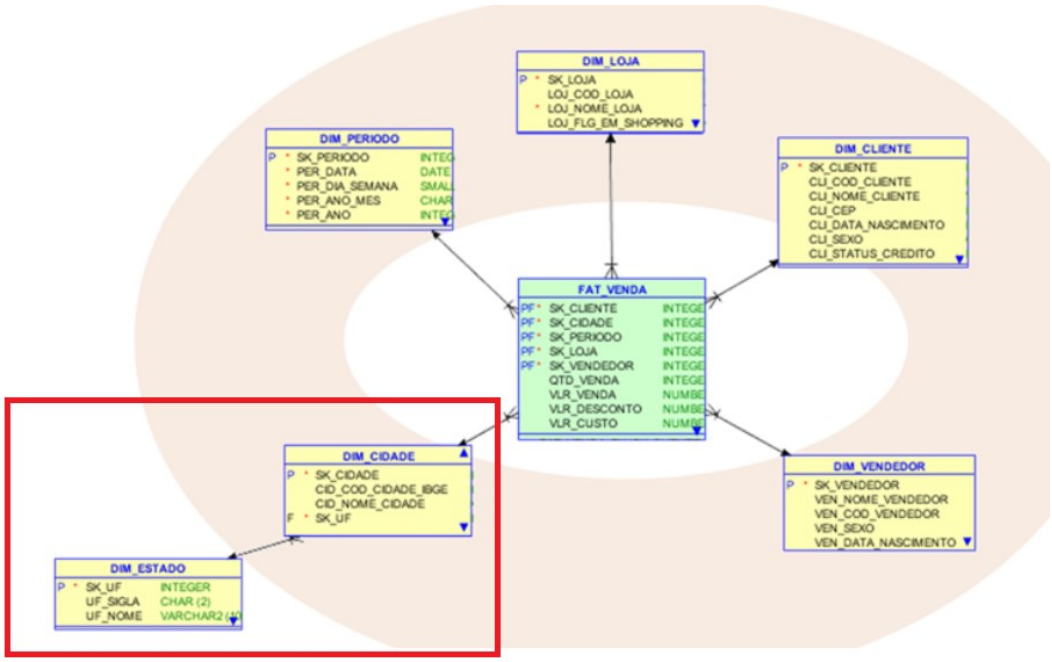
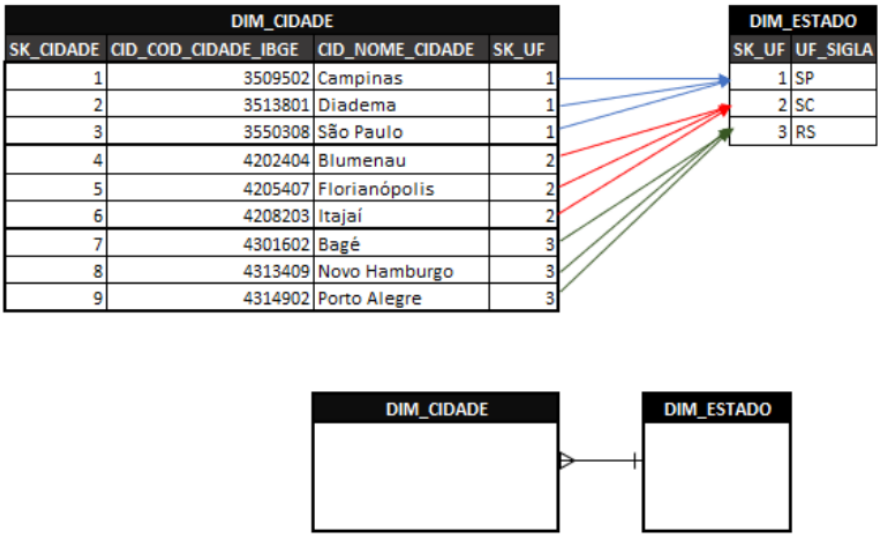


Figura 34 – Snowflake exemplo Cidade/Estado.



DIM_CIDADE

SK_CIDADE	CID_COD_CIDADE_IBGE	CID_NOME_CIDADE	SK_UF
1	3509502	Campinas	1
2	3513801	Diadema	1
3	3550308	São Paulo	1
4	4202404	Blumenau	2
5	4205407	Florianópolis	2
6	4208203	Itajaí	2
7	4301602	Bagé	3
8	4313409	Novo Hamburgo	3
9	4314902	Porto Alegre	3

DIM_ESTADO

SK_UF	UF_SIGLA
1	SP
2	SC
3	RS

É possível perceber que com o novo modelo criamos a nova dimensão Estado e levamos somente a chave de Estado para a dimensão cidade, não sendo necessário repetir o nome do estado para todas as cidades de um mesmo estado. Neste cenário pode parecer que a redundância faria pouca diferença quanto ao espaço utilizado, porque ao invés de repetir SP, MG, estamos usando a chave. Mas dependendo do tipo do dado e do número de colunas que se repetem, a redundância pode influenciar muito na quantidade de espaço requerido.

Outro exemplo:

Figura 35 – Star Schema – Novo Exemplo.

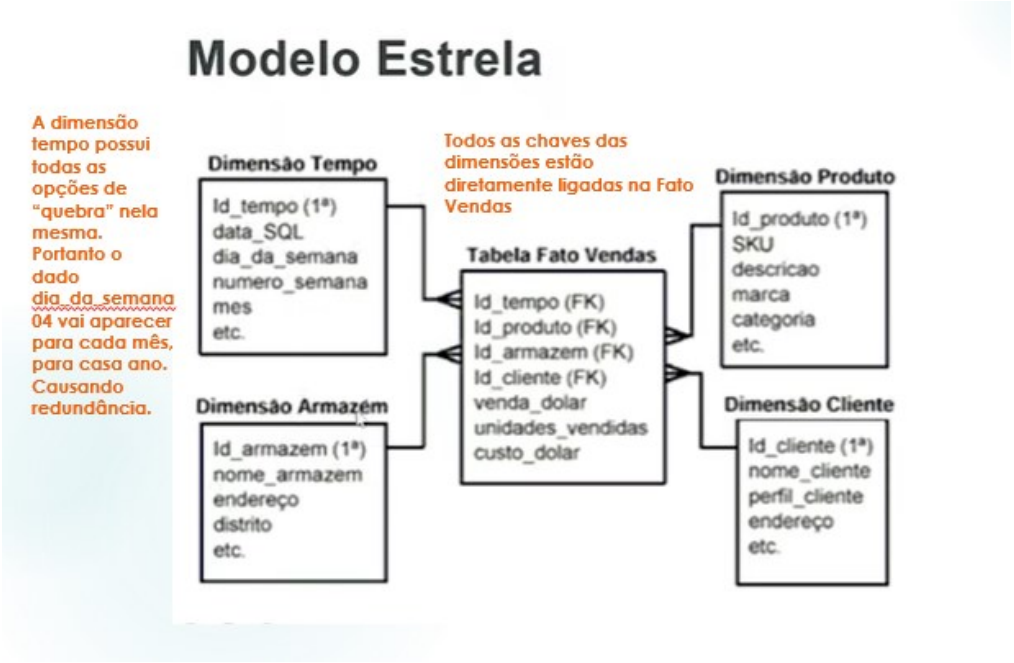
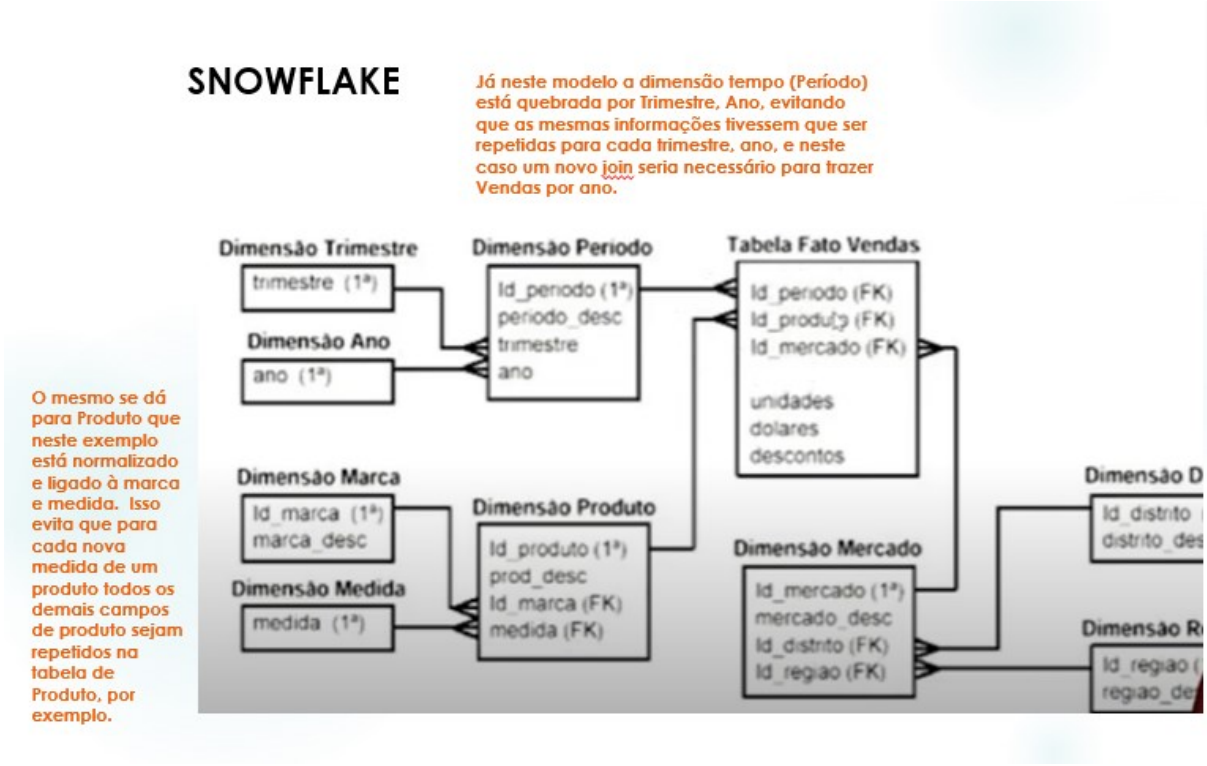


Figura 36 – Snowflake – Novo Exemplo.



4.4. ETL

ETL (Extract Transform Load) – ETC em Português (Extrair, Transformar e Carregar), é a fase do processamento de dados em que carregamos dados de diversas fontes, transformando os padrões em um formato único, e carregamos no ambiente de análise de dados, comumente DW.

4.5. OLAP (Processamento Analítico On-line)

OLAP é o termo usado para descrever um conjunto de técnicas para a análise de dados multidimensionais complexos do Data Warehouse ou de outras fontes como Data Lake, que será estudado em outros módulos do curso. Em resumo o DW é o armazenamento e o OLAP é o processamento destes dados em múltiplas visões. No

mercado existem diversas ferramentas de OLAP, dentre as quais podemos citar: IBM Cognos, Oracle OLAP, Oracle Essbase, Analysis Services do Microsoft SQL Server (SSAS).

4.6. Arquiteturas OLAP

A arquitetura OLAP pode ser implementada de diversas formas, de acordo com a necessidade do negócio, e isso vai definir as ferramentas que podem ser usadas, a velocidade de acesso aos dados, a flexibilidade das análises, o espaço requerido, dentre outros aspectos. As principais arquiteturas são:

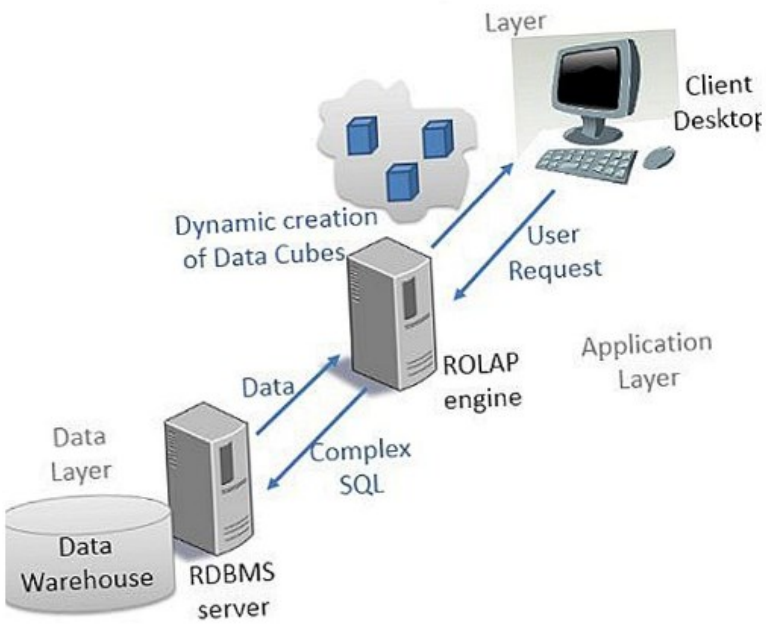
ROLAP:

ROLAP (Relational Online Analytical Processing) é um modelo no qual os dados são armazenados como no banco de dados relacional, com linhas e colunas e para exibir os dados em uma visualização multidimensional é criada uma camada semântica de metadados que mapeia as dimensões para as tabelas relacionais. Metadados também suportam agregação dos dados.

Sempre que o mecanismo ROLAP no servidor analítico emite uma consulta complexa, ele busca dados do warehouse principal e cria dinamicamente uma exibição multidimensional de dados para o usuário. Aqui ele se difere do MOLAP porque o MOLAP já possui uma visualização multidimensional pré-processada estática salva nos bancos de dados proprietários MDDBs

Como a visão multidimensional dos dados é criada dinamicamente ela é processada mais lentamente em comparação com o MOLAP. O mecanismo ROLAP lida com grandes volumes de dados.

Figura 37 – ROLAP.



MOLAP:

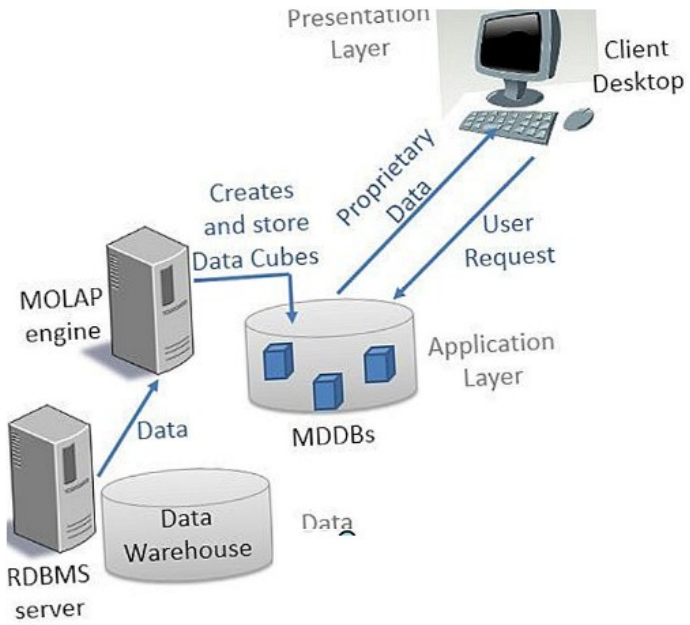
MOLAP é um modelo Multidimensional Online Analytical Processing. Os dados utilizados para análise são armazenados em bancos de dados multidimensionais especializados (MDDBs). Os sistemas multidimensionais de gerenciamento de banco de dados são sistemas de softwares proprietários.

As células ou cubos de dados deste banco de dados possuem os dados pré-calculados e pré-fabricados gerados a partir do DW. Os sistemas de software proprietários criam esses dados pré-calculados ao solicitarmos o processamento dos cubos e os salvam nos bancos MDDBs mencionados anteriormente.

Agora é o trabalho do mecanismo MOLAP, que reside na camada do aplicativo, fornecer a visualização multidimensional dos dados dos MDDBs para o usuário. Assim, quando chega uma solicitação do usuário para os dados, nenhum tempo é desperdiçado

no cálculo dos dados e o processo é muito rápido. Uma linguagem usada para manipular os dados dos cubos é o MDX (Expressões Multidimensionais).

Figura 38 – ROLAP.



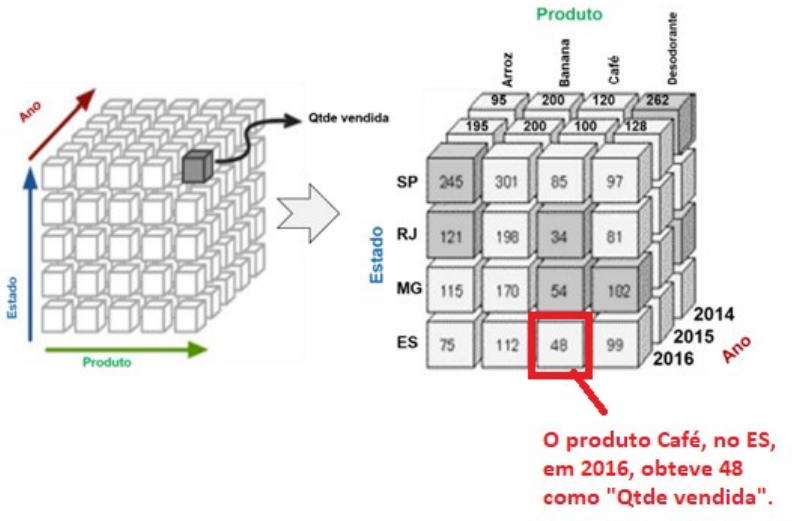
E temos ainda a arquitetura Híbrida, a HOLAP, que mistura os dois conceitos, WOLAP (Web), que utiliza um navegador Web para disparar a consulta no cubo, SOLAP, voltada para cubos geográficos e DOLAP (Desktop), ferramentas que geram cubos localmente. Mas as principais arquiteturas são ROLAP e MOLAP.

Conforme já mencionamos anteriormente, o termo Cubo é utilizado para designar o resultado de um processamento OLAP, por sua característica multidimensional, que em analogia ao cubo da matemática, nos permite analisar um fato sob diferentes “ângulos”, ou dimensões.

No exemplo abaixo temos um cubo criado por um sistema OLAP, em que métrica “Qtde vendida” (correspondente à métrica da sua fato Vendas), pode ser

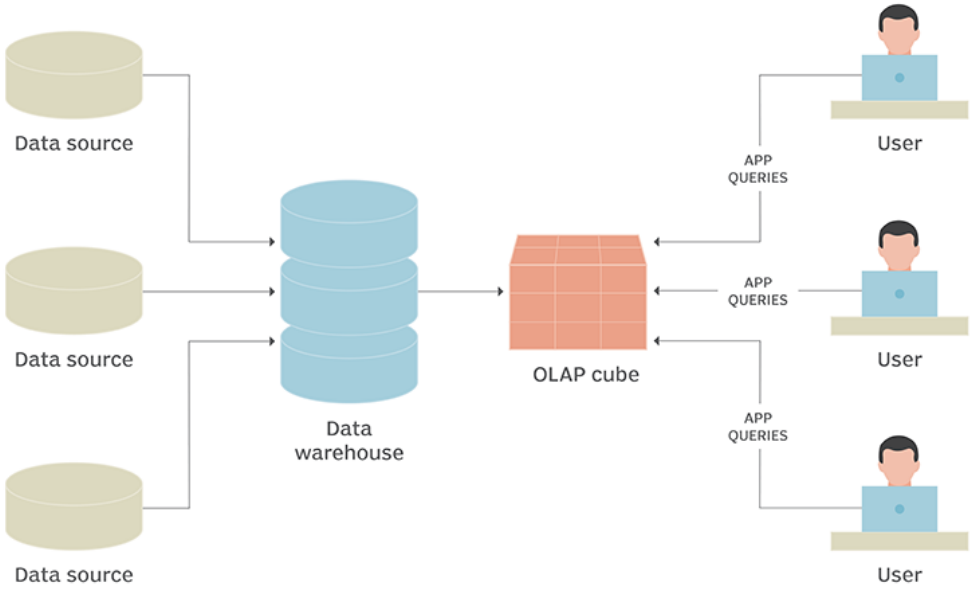
analisada por Estado, Ano ou Produto, ou mais de uma dessas dimensões combinadas. Portanto, podemos dizer analisando o segundo Cubo da imagem, que em 2016 o produto Café vendeu 48 unidades.

Figura 39 – Cubo.



Na Figura abaixo podemos perceber que os cubos são criados para disponibilizar visões ainda mais complexas sobre os dados armazenados nos DW.

Figura 40 – Cubos análise DW.



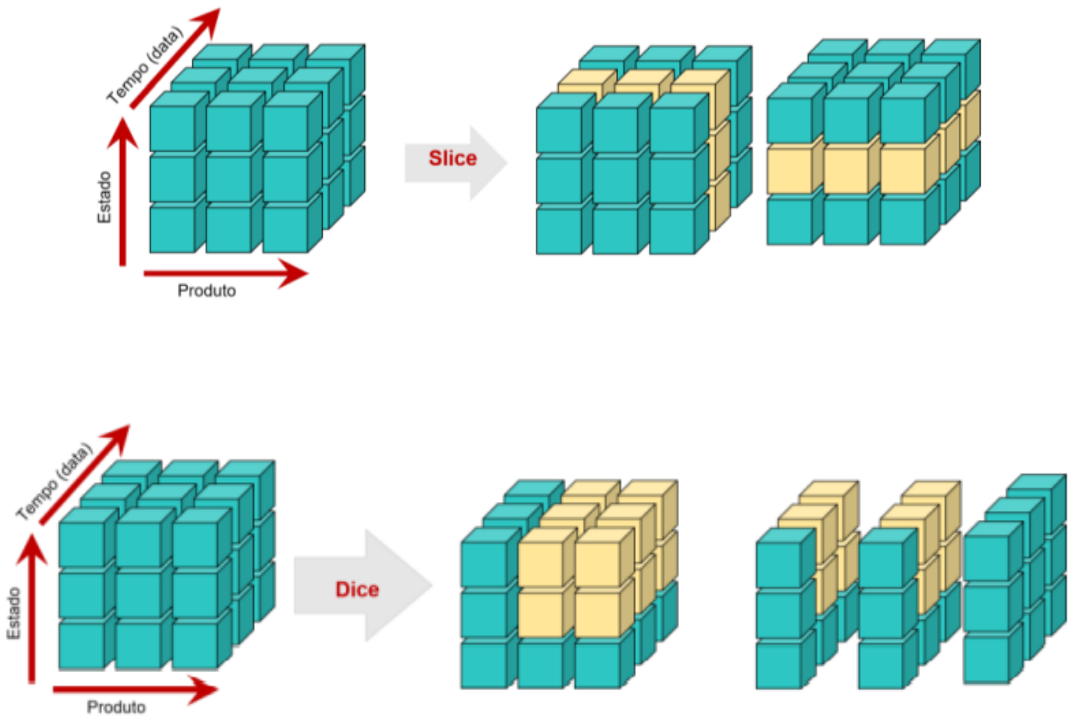
Nos cubos os usuários podem realizar operações que resultam em análises profundas sobre o negócio de maneira extremamente rápidas, pois os cubos em geral são pré-processados e todo o conteúdo do cubo já está disponível para consulta. As principais operações são drill down, roll-up, pivot, slice e dice, elas permitem interagir e manipular dados multidimensionais.

Drill down: também chamada de agregação ou consolidação, em que durante a análise de um cubo vamos de algo menos detalhado para algo mais detalhado. Por exemplo, estamos analisando vendas por Estado inicialmente e passamos a analisar por Cidade. Dessa maneira, realizamos a operação de Drill down.

Já o Roll up é o oposto do Drill down, pois passamos de uma análise mais detalhada para menos detalhada. Seria, ainda utilizando o exemplo de Cidade/Estado, eu passar de uma análise por Cidades para agrupamentos por Estado.

Também temos as operações de Slice e Dice, também conhecidas como seleção e projeção, que é como se reduzíssemos o cubo gerando cubos menores para análise de parte dos dados. No Slice filtramos os valores das dimensões para só trazer os dados que queremos, por exemplo: só quero analisar pelo estado de MG, então removo do filtro os demais estados. Enquanto no Dice, eu removo uma ou mais dimensões e analiso pelas demais, por exemplo: quero analisar vendas por ano, não por Estado, portanto removo essa dimensão no resultado da minha análise. Na Figura abaixo podemos entender melhor:

Figura 41 – Slice e Dice.



Por fim, na operação de Pivot, invertemos a ordem de análise modificando a ordem das dimensões, por exemplo: ao invés de analisar vendas por Produto x Estado,

quero ter a visão primeiro do total por Estado e então quebrar por Estado (Estado x Produto).

Entendidos os conceitos fundamentais da Engenharia de Dados é hora de seguir adiante na caminhada para descobrir mais sobre esse universo que permeia todas as áreas e que tem cada dia mais se mostrado crucial como diferencial competitivo das empresas!

Aproveitem o caminho e bons estudos!

Referências

BATINI, C.; CERI, S.; NAVATHE, S. **Conceptual Database Design**: An Entity-Relationship Approach. Redwood City: Addison-Wesley, 1992.

DAVENPORT, T.; PRUSAK, L. **Conhecimento Empresarial**: como as organizações gerenciam o seu capital intelectual. Rio de Janeiro: Elsevier, 2003.

FERREIRA, J. E. **Banco de Dados**: Modelo Entidade - Relacionamento. Disponível em: https://www.ime.usp.br/~jef/bd02#:~:text=MER%3A%20Conjunto%20de%20conceitos%20e,dados%20que%20conhece%20o%20MER.&text=representa%20%C3%A9%20a%20entidade. Acessado em: 30 jun, 2022.

FRIEDMAN, U. **Big Data**: A short history. 2012. Disponível em: https://foreignpolicy.com/2012/10/08/big-data-a-short-history/. Acessado em: 30 jun, 2022.

GOMES, P. C. T. **O que faz um engenheiro de dados?**. 2019. Disponível em: https://www.datageeks.com.br/engenheiro-de-dados/. Acessado em: 30 jun. 2022.

HENRIQUE, K. **A Linguagem SQL para SEFAZ ES**: Consultas e Operadores Condicionais. 2021. Disponível em: https://www.estrategiaconcursos.com.br/blog/linguagem-sql-para-sefaz-es/. Acessado em: 30 jun. 2022.

HENRIQUE, K. **Aprenda sobre Dado, Informação, Conhecimento e Inteligência em TI**. 2021. Disponível em: https://www.estrategiaconcursos.com.br/blog/dado-informacao-conhecimento-inteligencia/. Acessado em: 30 jun, 2022.

NAVATHE, S.; ELMASRI, R. **Sistemas de Banco de Dados**. 6. ed. São Paulo: Pearson Universidades. 2011. p. 03.

SALESFORCE BLOG. **Data Warehouse e Data Lake**: Quais as diferenças? 2020. Disponível em: https://www.salesforce.com/br/blog/2020/10/data-warehouse-e-data-lake.html. Acessado em: 30 jun, 2022.

SANCHES, A. **Disciplina: Fundamentos de Armazenamento e Manipulação de Dados. Aula: Modelo Físico**. 2005. Disponível em: https://www.ime.usp.br/~andrers/aulas/bd2005-1/aula12.html. Acessado em: 30 jun. 2022.

SENE, A. **O que faz um Engenheiro de Dados?**. 2018. Disponível em: https://medium.com/data-hackers/o-que-faz-um-engenheiro-de-dados-fdcb0bca966b. Acessado em: Acessado em: 30 jun, 2022.

SILBERSCHATZ, A.; KORTH, H. F.; SUDARSHAN, S. **Database System Concepts**. McGraw-Hill Education, 2019.

SILVA, A. **Qual a diferença entre análise descritiva, preditiva e prescritiva?**. 2021. Disponível em: https://operdata.com.br/blog/qual-a-diferenca-entre-analise-descritiva-preditiva-e-prescritiva/#:~:text=A%20an%C3%A1lise%20descritiva%20%C3%A9%20respons%C3%A1vel,de%20acordo%20com%20cada%20cen%C3%A1rio. Acessado em: 30 de jun. 2022.

ZONTA, R. **Fatos e Dimensões II**. 2020. Disponível em: https://medium.com/@ricardozontasantos/fatos-e-dimens%C3%B5es-ii-a74cde5ab70. Acessado em: 30 jun, 2022.