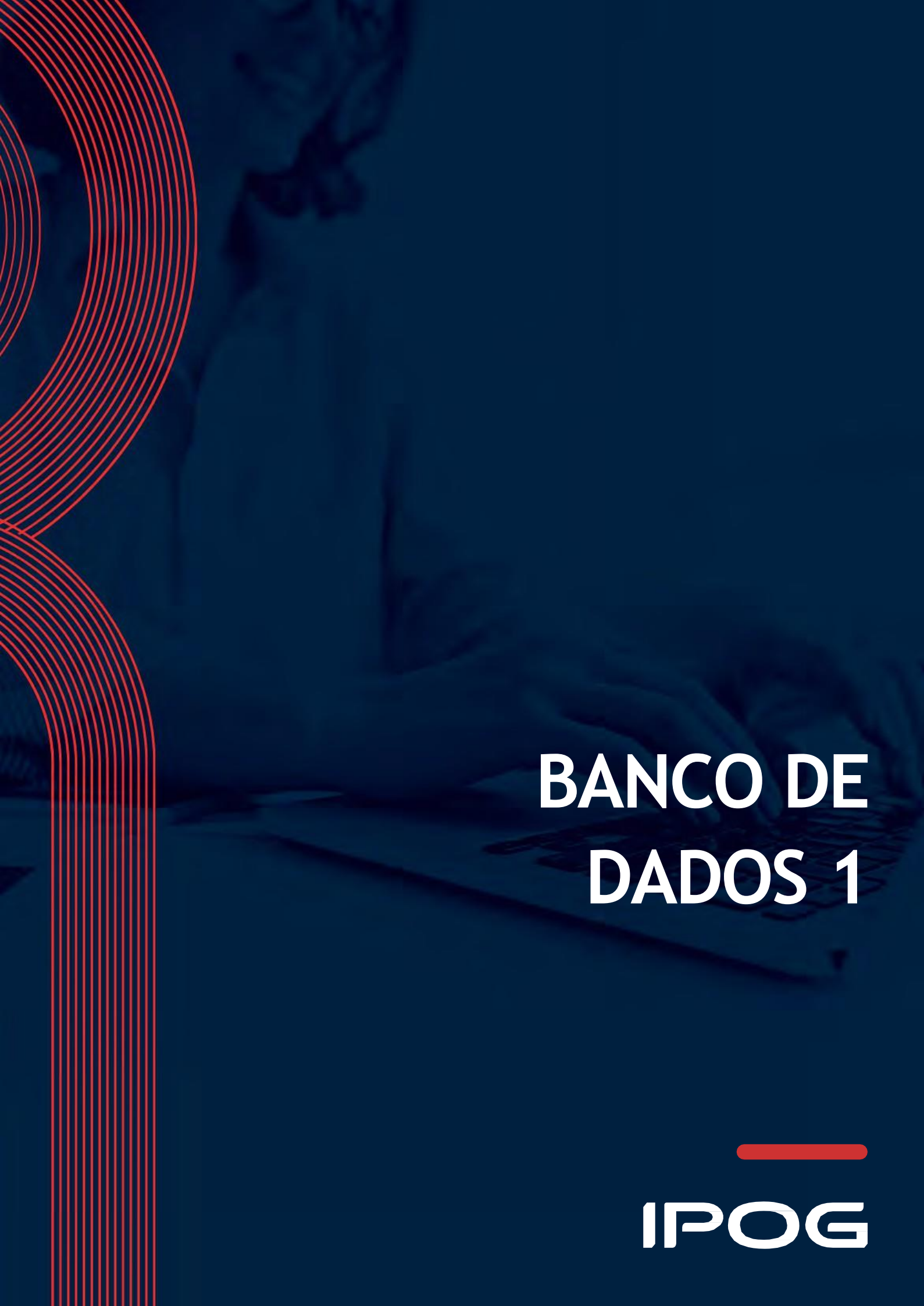


BANCO DE DADOS 1

IPOG

EDITORA IPOG

IP5p Instituto de Pós-Graduação e Graduação – IPOG

Banco de Dados 1. / Autor: Joelma de Moura Ferreira.

240f. :il.

ISBN:

1. Banco de Dados 2. Modelo de Dados 3. Modelo Conceitual 4. Modelo Lógico 5. Linguagem SQL.

CDU: 005

IPOG

Instituto de Pós Graduação e Graduação

Sede

Av. T-1 esquina com Av. T-55 N. 2.390 - Setor Bueno - Goiânia-GO.

www.ipog.edu.br | (62) 3945-5050

Sumário

APRESENTAÇÃO

Seja bem-vindo à disciplina de Banco de Dados 1.

Nesta jornada de aprendizado, vamos explorar os fundamentos essenciais que compõem o mundo fascinante dos bancos de dados. Desde a distinção entre dados, informação e banco de dados, até a importância crucial dos Sistemas Gerenciadores de Banco de Dados (SGBD), mergulharemos em aspectos fundamentais que moldam essa área vital da tecnologia da informação.

Na primeira unidade, começaremos com uma introdução aos conceitos de dados, informação e banco de dados. Entender esses conceitos é fundamental para construir uma base sólida de conhecimento em banco de dados e preparar o terreno para explorarmos tópicos mais avançados.

Em seguida, mergulharemos nos Sistemas Gerenciadores de Banco de Dados (SGBD), compreendendo não apenas o que são e qual sua função, mas também explorando suas principais características e funções, além de conhecer exemplos populares e suas aplicações.

Na sequência abordaremos a modelagem de dados. Exploraremos o Modelo de Dados, começando pela introdução ao Modelo Entidade-Relacionamento (MER). Vamos compreender como representar entidades, atributos e relacionamentos, permitindo-nos projetar bancos de dados eficientes e bem estruturados. A modelagem de dados é uma etapa crucial no desenvolvimento de sistemas de informação e desempenha um papel fundamental na garantia da integridade e

consistência dos dados.

Estou confiante de que esta jornada será enriquecedora e desafiadora, e estou animada para vê-lo explorar cada tópico com curiosidade e entusiasmo. Esteja preparado para desafiar suas ideias preconcebidas, expandir seus horizontes e criar uma base sólida para o seu crescimento como profissional de banco de dados. Vamos começar esta jornada juntos e aproveitar ao máximo. Prepare-se para uma experiência de aprendizado transformadora.

Boa leitura e estudos!

Joelma de Moura

OBJETIVOS

OBJETIVO GERAL

Adquirir entendimento sobre os princípios de bancos de dados e desenvolver habilidades para reconhecer e criar sistemas de banco de dados até o nível logico.

OBJETIVOS ESPECÍFICOS

- Conhecer e manipular sistemas gerenciadores de banco de dados.
- Compreender os modelos conceituais de banco de dados.
- Entender os fundamentos do modelo relacional.
- Projetar banco de dados relacional.
- Compreender os conceitos fundamentais de DQL (Data Query Language): SQL.

Conheça como esse conteúdo foi organizado

Unidade 1: Fundamentos de Banco de Dados;

Unidade 2: Modelo de Dados;

Unidade 3: Normalização;

Unidade 4: Modelo Físico;

Unidade 5: Manipulação de dados.

UNIDADE 1

FUNDAMENTOS DE BANCO DE DADOS

É com grande prazer que damos início a essa jornada de aprendizado e descobertas. Nesta unidade, teremos a oportunidade de explorar e aprofundar nossos conhecimentos em um tema fascinante.

Ao final dos estudos, você deverá ser capaz de: compreender a definição e os conceitos-chave de banco de dados e explorar a instalação do ambiente.

1.1 Introdução aos Conceitos de Dados, Informação e Banco de Dados

Dado, informação e conhecimento são termos inter-relacionados, mas com significados distintos no contexto da tecnologia.

Dado: Os dados são elementos brutos e desorganizados que representam fatos, números, símbolos ou caracteres que ainda não possuem contexto ou significado específico. Por si só, os dados são apenas pontos de dados isolados, como letras, números ou símbolos, que não têm significado ou utilidade. Por exemplo, "12", "vermelho" ou "A".

Informação: Informação é o dado que foi processado, organizado ou estruturado de forma a ter um contexto, significado e relevância. Ela fornece respostas a perguntas como: quem? o quê? Onde? Quando? e por quê? tornando os dados compreensíveis e utilizáveis. Por exemplo, se temos o dado "12" representando a temperatura, a informação seria "12 graus Celsius". A informação adiciona valor aos dados, permitindo que sejam compreendidos e utilizados para tomar decisões ou realizar ações.

Conhecimento: O conhecimento é um nível mais elevado de compreensão e compilação de informações. Ele vai além da simples interpretação dos dados e informações, envolvendo a

aplicação de entendimentos, experiências e insights para resolver problemas, tomar decisões e criar ideias. O conhecimento é construído ao longo do tempo através da experiência, aprendizado e reflexão sobre informações e experiências passadas. Por exemplo, o conhecimento médico envolve a aplicação de informações sobre anatomia, fisiologia e medicamentos para diagnosticar e tratar doenças.

Um banco de dados é uma coleção organizada de dados que são armazenados e gerenciados eletronicamente. Esses dados podem ser de vários tipos como texto, números, imagens, vídeos, entre outros formatos. O objetivo principal de um banco de dados é fornecer um meio eficiente para armazenar e recuperar informações de maneira rápida e precisa.

Os dados armazenados em um banco de dados são posteriormente processados por aplicações de software. Essas aplicações podem realizar uma variedade de operações nos dados como inserção, atualização, exclusão e consulta. Além disso, as aplicações podem executar análises e cálculos complexos nos dados para produzir informações significativas.

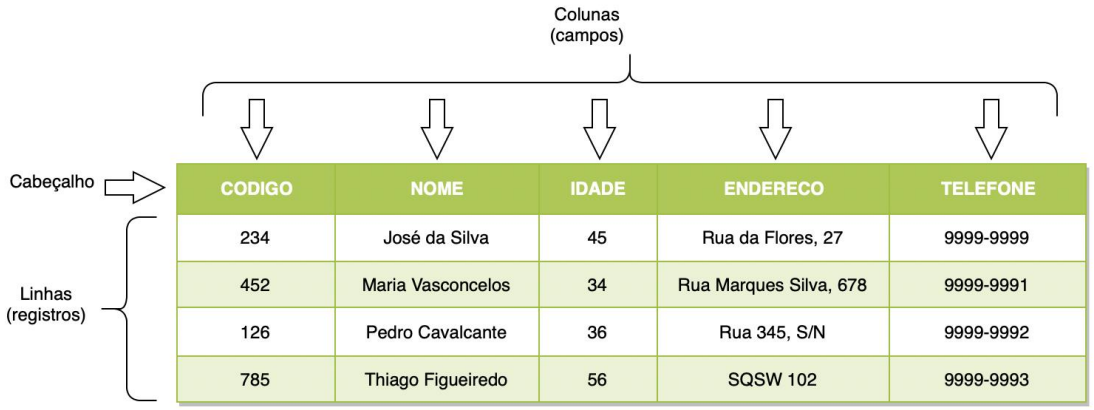
Essas informações são disponibilizadas aos usuários por meio de interfaces de usuário, relatórios, dashboards e outras formas de apresentação. Essas informações ajudam os usuários a tomar decisões informadas e embasadas nos dados disponíveis.

1.2 Importância dos Bancos de Dados da Organização e Manipulação de Informações

- **Tabela**

Uma tabela é uma estrutura compostas por linhas e colunas na qual armazenamos os dados.

Por exemplo, supondo que uma empresa queira armazenar dados como código, nome, endereço e telefone de quatro de seus clientes. Poderíamos criar uma tabela CLIENTE contendo colunas com CODIGO, NOME, ENDERECO, IDADE e TELEFONE. Nas linhas desta tabela escreveríamos os dados de cada cliente individualmente. Como seriam quatro clientes, a tabela teria um cabeçalho e quatro linhas de dados. Veja o exemplo ilustrado na Figura 1.1.



CODIGO	NOME	IDADE	ENDERECO	TELEFONE
234	José da Silva	45	Rua da Flores, 27	9999-9999
452	Maria Vasconcelos	34	Rua Marques Silva, 678	9999-9991
126	Pedro Cavalcante	36	Rua 345, S/N	9999-9992
785	Thiago Figueiredo	56	SQSW 102	9999-9993

Figura 1.1: Exemplo de uma tabela de dados cadastrais de clientes

Definimos nomes especiais para os elementos de uma tabela. As colunas chamamos CAMPOS, e as linhas chamamos REGISTROS. Perceba que na tabela, o campo "endereço" está sem o cedilha e o código sem o acento, isso é porque evitamos colocar acentos nos nomes dos campos e da própria tabela. Também não colocamos espaço, por exemplo, se quiser dar o nome Vendas do Mês para um campo, o ideal é não colocar espaço ou acento. Um bom nome seria VENDAS_MES, substituímos o espaço por _ e tiramos o acento.

Os nomes dos campos e da própria tabela são importantes, pois eles já dão uma ideia de quais dados estão armazenados. Procure sempre dar nomes condizentes com o conteúdo. Para garantir a consistência dos dados, busque criar tabelas cujos dados descrevam um elemento de forma única como: tabela dados dos ALUNOS, tabela das VENDAS, tabela dos PRODUTOS e assim por diante. Os dados que são armazenados em uma tabela devem descrever um elemento ou a relação entre elementos (como veremos mais adiante).

Exercício 1: VAMOS PRATICAR!!

Se você tivesse que projetar uma tabela para armazenar os dados que estão na sua carteira de identidade. Quais seriam os campos que colocaria? Dê nome a tabela, aos campos e coloque pelo menos dois registros (podem ser fictícios). Faça sem olhar a resposta. Depois, compare com a sua. Não existe uma resposta única. Os campos podem ficar com nomes diferentes, a ordem pode não ser a mesma. O importante é que siga a regra de nomenclatura dos campos.

Resposta Exercício 1

TB_CIDADAO

CPF	RG	NOME	MAE	Dt_NASCIMENTO	NATURALIDADE	DT_EXPEDICAO	ORGAO
89865456543	67656	José da Silva	Maria Aparecida	12/07/1978	São Paulo	30//09/2022	SSPSP
87564343219	234234	Thiago Figueiredo	Helena Dias	29/08/1967	Goiânia	23/04/1989	SSPGO

- **Tipo dos campos**

Um outro ponto importante no momento de criar uma tabela de um banco de dados no computador é definir o tipo de dados que será armazenado em cada coluna. O dado pode ser texto, número, data ou campo do tipo verdadeiro ou falso. Definimos também o tamanho máximo de cada coluna.

Para o exemplo da tabela CLIENTES, podemos definir:

- CODIGO: texto com tamanho de 3 caracteres no máximo
- NOME: texto com tamanho de 50 caracteres no máximo
- IDADE: valor inteiro
- ENDERECO: texto com tamanho de 200 caracteres no máximo
- TELEFONE: texto com tamanho de 9 caracteres no máximo

Isso é muito importante para controle e tratamento dos dados. Não sabe o que é um tipo de dados?

pesquise em: https://pt.wikipedia.org/wiki/Tipo_de_dado

Exercício 2: VAMOS PRATICAR!!

Quais seriam os tipos de dados e tamanho da tabela criada por você no Exercício 1?
Resposta do Exercício 2:

- CPF: texto com tamanho de 11 caracteres no máximo
- RG: texto com tamanho de 15 caracteres no máximo
- NOME: texto com tamanho de 50 caracteres no máximo
- MAE: texto com tamanho de 50 caracteres no máximo
- DT_NASCIMENTO: data
- NATURALIDADE: texto com tamanho de 20 caracteres no máximo
- DT_EXPEDICAO: data
- ORGAO: texto com tamanho de 50 caracteres no máximo

- **Banco de Dados**

Um banco de dados pode ser definido de forma "simples" como um conjunto de tabelas. ***Mas, não é só isso. Mais na frente aprimoraremos essa definição***. Por enquanto, ficaremos com essa visão simplificada. Um banco de dados simples conteria uma única tabela, quanto mais tabelas, mais complexo é o banco de dados. Já dá para imaginar que os sistemas de uma empresa lidam com dezenas e centenas de tabelas diferentes.

Vamos voltar para a empresa que quer armazenar dados dos seus clientes. Imaginemos que agora ela queira armazenar também dados dos vendedores que fazem parte do quadro de colaboradores. Seriam apenas três vendedores e os campos a serem usados são MATRICULA, NOME, TELEFONE. O conjunto das duas tabelas e dos dados armazenados nela, por enquanto, irão compor o nosso banco de dados. A Figura 1.2 ilustra esse exemplo.

CLIENTES

CODIGO	NOME	IDADE	ENDERECO	TELEFONE
234	José da Silva	45	Rua da Flores, 27	9999-9999
452	Maria Vasconcelos	34	Rua Marques Silva, 678	9999-9991
126	Pedro Cavalcante	36	Rua 345, S/N	9999-9992
785	Thiago Figueiredo	56	SC	

VENDEDORES

MATRICULA	NOME	TELEFONE
2335	Tomé Mascarenhas	9999-8888
3426	André Costa	9999-8882
2421	João Sousa	9999-8883

Figura 1.2: Exemplo das duas tabelas do banco de dados: clientes e vendedores

- **Banco de dados relacional**

Até agora, vimos o que são dados e como recuperá-los, que podemos armazenar dados em estruturas chamadas tabelas, dividindo esses dados em campos, campos esses que possuem um tipo e tamanho máximo. As tabelas também possuem linhas que são os registros individualizados do conjunto de dados (Figura 1.3).

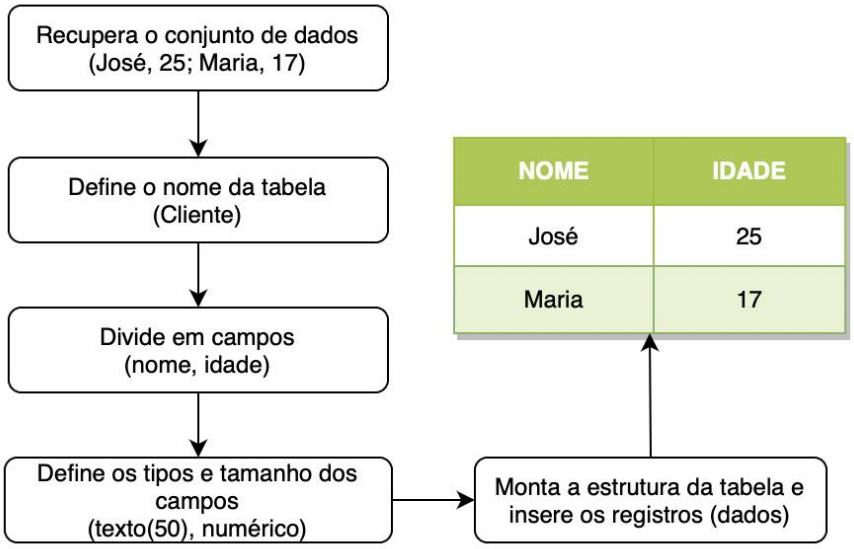


Figura 1.3: Fluxo da definição da tabela

Como um banco de dados pode ter várias tabelas, o fluxo ilustrado na Figura 1.3 deve ser repetido para cada tabela que pretendemos criar.

- **A importância de banco de dados**

Esses são alguns dos principais aspectos que destacam a importância dos bancos de dados.

- Organização Estruturada: os dados são organizados, permitem relacionamentos entre diferentes conjuntos de dados e facilitam a análise e recuperação de informações específicas.
- Armazenamento Centralizado: os bancos de dados fornecem um local centralizado para armazenar grandes volumes de dados de forma estruturada.
- Facilidade de Acesso: permitem o acesso rápido e eficiente às informações por meio de consultas SQL (*Structured Query Language*) ou interfaces de programação.
- Integridade dos Dados: os bancos de dados garantem a integridade dos dados por meio de restrições e validações, evitando a inserção de informações inconsistentes ou inválidas.
- Segurança dos Dados: os sistemas de gerenciamento de banco de dados (SGBDs) oferecem recursos de segurança, como autenticação de usuários e controle de acesso, garantindo a proteção dos dados contra acesso não autorizado.

- Backup e Recuperação: permitem a realização de backups regulares dos dados, garantindo a recuperação rápida em caso de falhas de hardware, erros humanos ou desastres naturais.
- Escalabilidade: os sistemas de banco de dados são projetados para lidar com grandes volumes de dados e podem ser dimensionados conforme necessário para atender aos requisitos de crescimento da organização.
- Suporte a Transações: os bancos de dados oferecem suporte a transações, garantindo consistência e atomicidade durante operações de inserção, atualização e exclusão de dados.
- Análise de Dados: permitem a execução de consultas complexas e análises de dados para obter insights valiosos e tomar decisões informadas.
- Integração de Aplicações: os bancos de dados podem ser integrados a diversas aplicações, permitindo o compartilhamento de dados entre diferentes sistemas e aumentando a eficiência operacional da organização.

Mas o que são sistemas de gerenciamento de Banco de Dados (SGBDs)?

1.3 Sistemas Gerenciadores de Banco de Dados (SGBD)

1.3.1 O Que é um SGBD e a sua Função?

Um Sistema de Gerenciamento de Banco de Dados (SGBD) é composto por uma série de programas dedicados à administração da estrutura de um banco de dados e à regulação do acesso aos dados nele contidos.

Ele atua como uma interface entre o usuário e o banco de dados, garantindo a gestão eficiente das informações. A estrutura do banco de dados é mantida por exemplo, em arquivos, sendo o SGBD o único meio de acesso a esses dados.

As funcionalidades do SGBD abrangem desde a criação e modificação de estruturas, realização de testes, backups e restauração de dados, até a análise de desempenho e outras tarefas relacionadas à gestão eficiente do sistema.

Exemplos: MySql, Postgresql, Oracle, Sql Server, Mongo, Db2, Caché, Firebird, Hsql

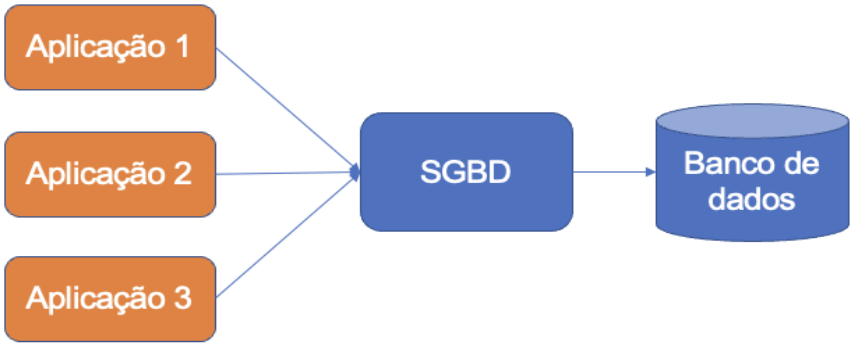


Figura 1.4: Conexão SGBD

1.3.2 Principais Características e Funções de um SGBD

São vantagens do uso de SGBDs:

- Controle de redundância.
- Aprimoramento da segurança de dados com o controle de acesso e uso de restrições e privilégios.
- Armazenamento persistente.
- Backup e Recuperação.
- Compartilhamento de dados.
- Fornecimento de múltiplas interfaces do usuário atendendo os usuários com diferentes níveis de conhecimento.
- Gerenciamento de transações.
- Restrições de integridade.

Propriedades fundamentais

Consistência de dados refere-se à garantia de que os dados armazenados no banco de dados estarão sempre corretos e coerentes. Isso significa que as informações não podem entrar em conflito umas com as outras e devem seguir regras de integridade definidas. Na Figura 1.5, vemos a mesma informação NOME nas tabelas, mas o dado em si, em cada tabela, está diferente.

Nome	Endereço	Telefone
João Lemos Silva	Rua A	3867-3789
Maria de Vasconcelos	Rua B	6735-8823
José de Almeida Sales	Rua C	2789-3790

Nome	Profissão	Idade
João L. Silva	Geólogo	32
Maria Vasconcelos	Professora	45
José Sales	Advogado	56

Figura 1.5: Exemplo de uma tabela de dados

A **completeza** diz respeito à garantia de que todas as informações relevantes são devidamente registradas no banco de dados. Isso significa que o banco de dados deve conter todos os dados necessários para atender aos requisitos do sistema ou aplicação, permanecendo completo e atualizado em todos os momentos. Na figura 6, vemos que ENDERECO tem registros com dados faltantes. Se esse é um campo importante, temos um problema de completeza.

Nome	Endereço	Telefone
João	Rua A	3867-3789
Maria		6735-8823
José	Rua C	2789-3790

Figura 1.6: Exemplo de uma tabela de dados

A **validade** refere-se à garantia de que os dados armazenados no banco de dados estão corretos e representam com precisão o mundo real. Isso implica que os dados devem ser precisos, confiáveis e atualizados. Na figura 1.7, vemos que o campo TELEFONE tem um valor não válido.

Nome	Endereço	Telefone
João	Rua A	3867-3789
Maria	Rua B	JOELMA
José	Rua C	2789-3790

Figura 1.7: Exemplo de uma tabela de dados

1.3.3 Exemplos de SGBDS Populares e suas Aplicações

Existem diversos Sistemas de Gerenciamento de Banco de Dados (SGBDs) populares, cada um com suas características e aplicações específicas.

MySQL

Amplamente utilizado em uma variedade de ambientes, desde pequenos websites até grandes corporações. É especialmente popular para aplicações da web devido à sua facilidade de uso e desempenho confiável.

Microsoft SQL Server

Amplamente utilizado em ambientes corporativos, especialmente em empresas que utilizam outros produtos da Microsoft. É conhecido por sua escalabilidade e robustez, sendo frequentemente usado para aplicações empresariais críticas.

Oracle Database

Usado principalmente em grandes empresas e organizações que requerem um alto desempenho e escalabilidade. É comumente utilizado em aplicações empresariais de missão crítica como sistemas bancários, telecomunicações e empresas de serviços financeiros, onde a confiabilidade e a segurança são fundamentais.

PostgreSQL

Conhecido por sua robustez, conformidade com padrões e recursos avançados, o PostgreSQL é amplamente utilizado em uma variedade de aplicações.

MongoDB

É um banco de dados NoSQL orientado a documentos, o MongoDB é frequentemente escolhido para aplicações que requerem flexibilidade no esquema de dados e alta escalabilidade como aplicações de comércio eletrônico, redes sociais e sistemas de gerenciamento de conteúdo.

Instalando o Mysql

Nesta disciplina, vamos usar o MySQL como nosso SGBD, para instalá-lo em ambiente Windows, entre no site, baixe o instalador da versão MySQL Community Server e o execute. https://dev.mysql.com/downloads/installer/

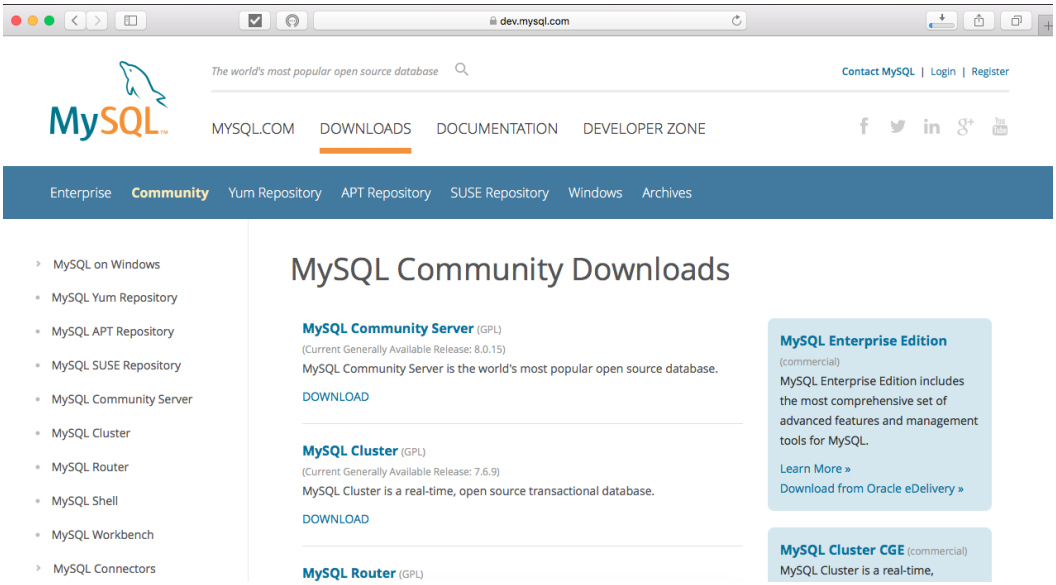


Figura 1.8: Baixando MySQL

Precisaremos também de uma interface para nos ajudar a manipular o banco de dados. Usaremos o MySQL Workbench. Para instalá-lo em ambiente Windows, entre no site, baixe o instalador e o execute. Faça isso depois de ter instalado o MySQL.

https://dev.mysql.com/downloads/workbench/

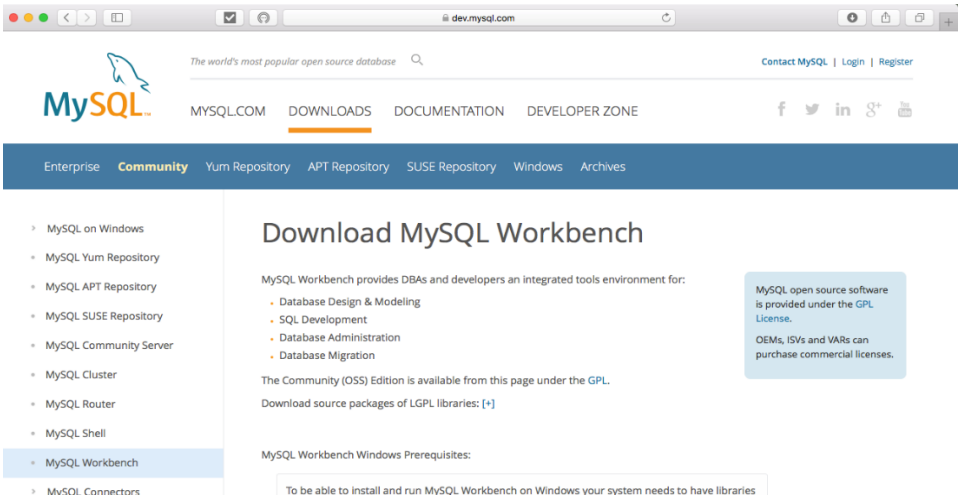


Figura 1.9: Baixando MySQL Workbench

Ambiente online de execução do Mysql

Caso não seja possível instalar o ambiente na máquina, podemos usar uma versão online de um SGBD. Execute o código MySQL usando o IDE on-line do myCompiler.

https://www.mycompiler.io/pt/new/mysql

Mas, tenha em mente que os recursos serão limitados, e nas aulas da disciplina será utilizado a versão Windows.

Ambiente online para criar os modelos

Vamos criar muitos modelos nesta disciplina e precisamos de uma ferramenta que nos auxilie a desenhar esses modelos. Vamos usar a versão online do brModelo. Entre no site e crie uma conta.

https://app.brmodeloweb.com/

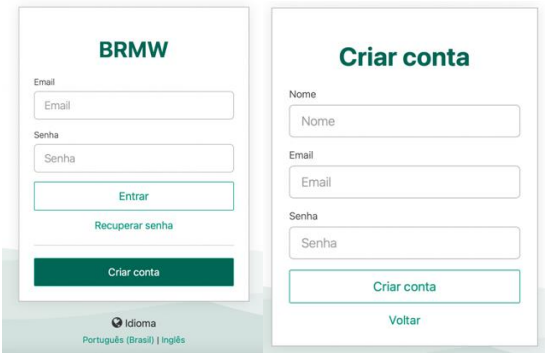


Figura 1.10: Login BRMW

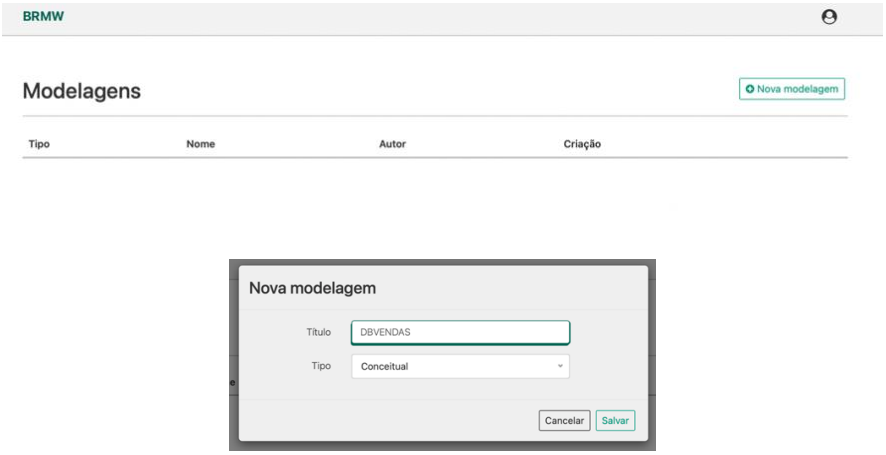


Figura 1.11: Criação de um modelo

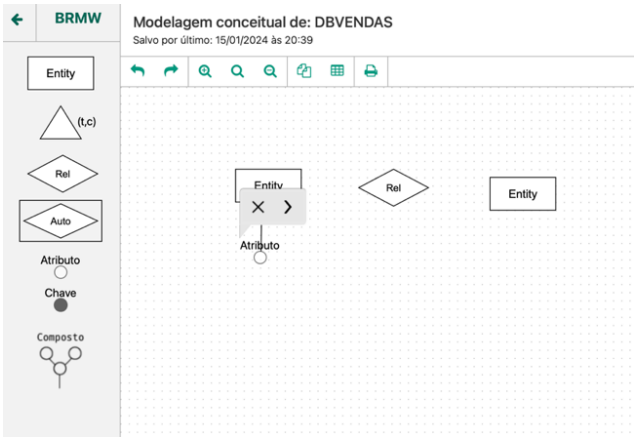


Figura 1.12: Área de trabalho brModelo We

UNIDADE 2

MODELO DE DADOS

Do que foi exposto até agora, percebemos que não é simples definir as tabelas, estruturar os tipos relacionamentos e tudo que está relacionado ao armazenamento de dados em um banco de dados. Então, **ANTES DE CRIAR AS TABELAS** é preciso seguir procedimentos que ajudarão a garantir que os dados serão mapeados da forma correta.

A primeira etapa é criar um modelo de dados, também conhecido como esquema de banco de dados. Esse modelo descreve de maneira formal a estrutura do banco de dados. Ele descreve quais os dados que o banco armazenará, não os dados em si, mas o tipo de dados. Para o nosso exemplo da empresa, um modelo de dados descreveria que armazenaríamos matrícula, nome e telefone dos vendedores.

O modelo de dados tem duas subcategorias: modelo conceitual e modelo lógico. O modelo conceitual independe do SGBD que será usado. Já o modelo lógico leva em consideração o tipo de SGBD que será usado. Temos ainda, o projeto ou modelo físico, que é a etapa final para construção real do banco de dados.

Ao final dos estudos, você deverá ser capaz de: entender sobre o processo de criação do modelo de dados, as características de um modelo conceitual e um modelo lógico.

2.1 Introdução ao Modelo Entidade-Relacionamento (MER)

- **Modelo conceitual**

Este modelo é criado na primeira fase de um projeto de banco de dados na qual os dados da empresa são mapeados e representados na forma de um Diagrama de Entidade Relacionamento (DER). Neste momento, não nos preocupamos se vamos criar tabelas ou não. Apenas modelamos quais dados a empresa necessita que sejam armazenados e tratados por uma aplicação de software.

- **Entidade**

Como todo diagrama, o DER possui elementos com finalidades específicas. A entidade é o elemento principal de um DER, representa o elemento da realidade que se deseja manter a informação.

Considere o cenário de uma organizadora de torneio de futebol que deseja organizar os dados dos torneios em uma base de dados. Sobre o que essa empresa precisaria manter informação? Pense nos SUBSTANTIVOS. Ela precisa por exemplo, manter informação dos TIMES, TORNEIOS, JOGADORES, PARTIDAS, ARBITROS. Essas informações seriam as entidades desse projeto de torneio.

E para um Petshop que deseja manter os dados dos atendimentos de animais que vão tomar banho, fazer tosa, etc.? Um bom conjunto de entidades seria: ANIMAIS, CLIENTES, ATENDIMENTOS, FUNCIONÁRIOS, PRODUTOS.

Podemos montar um conjunto de entidades para o nosso exemplo anterior de vendas. Seriam entidades: CLIENTES, VENDAS, VENDEDORES, PRODUTOS, ESTOQUE.

No DER uma entidade é representada por um retângulo. Na Figura 13 vemos seis entidades: CLIENTES, VENDAS, VENDEDORES, PRODUTOS, ESTOQUE e FUNCIONÁRIOS.

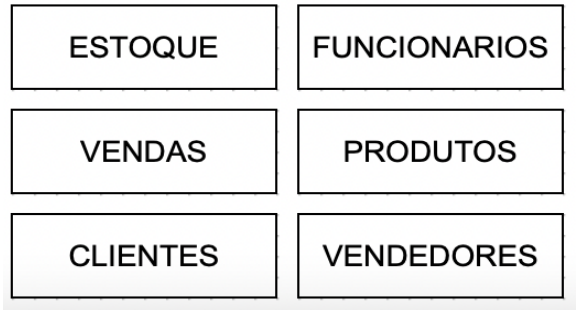


Figura 2.1: Exemplo de entidades de um sistema de venda

- **Atributo**

Uma propriedade de uma entidade é o atributo. Atributo é o dado que é associado a cada ocorrência de uma entidade. Por exemplo, se temos uma ocorrência de uma entidade PRODUTOS, isso significa que temos UM produto específico. Quais seriam os dados deste produto? Poderia ser CODIGO, NOME, PRECO. Esses seriam os atributos da entidade PRODUTOS.

Lembre-se, não interessa agora quais são os dados reais, só precisamos saber que será necessário armazenar os dados CODIGO, NOME, PRECO das instâncias da entidade

PRODUTOS. Os atributos podem ou não ser representados no DER. A representação seria:

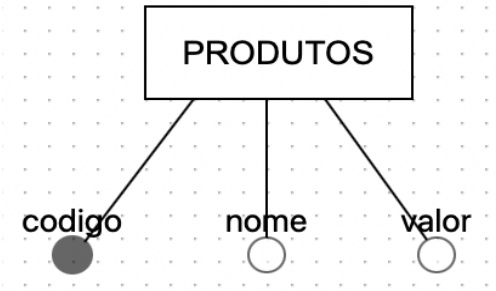


Figura 2.2: Representação gráfica dos atributos de uma entidade

Perceba que o código está pintado. Isso acontece porque se pensarmos nos códigos dos produtos, nunca existirá dois produtos com o mesmo código, pelo menos não deveria. Esse atributo é chamado de chave. Ele representa unicamente os valores do produto. Podemos ter produtos com o mesmo nome, com o mesmo valor, por isso, esses atributos não estão marcados, mas nunca com o mesmo código. Uma entidade dever ter pelo menos um atributo identificador chave, pois é ele que vai distinguir as ocorrências da entidade.

Vamos pensar no problema da empresa que precisa armazenar dos seus clientes e dos vendedores.

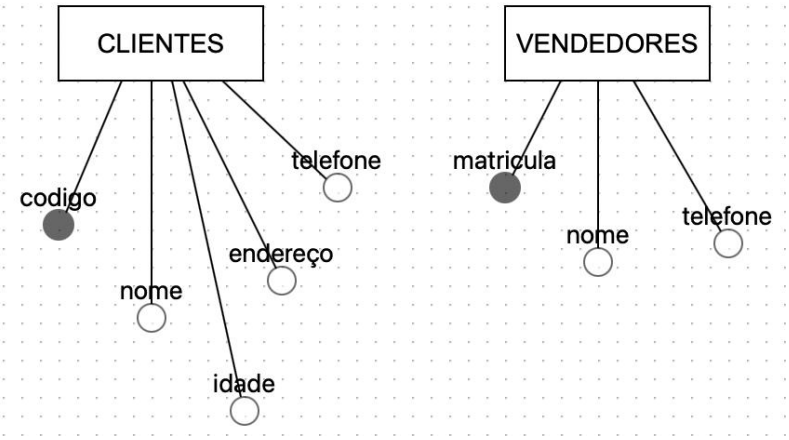


Figura 2.3: Entidade do problema das vendas

Exercício 1: VAMOS PRATICAR!

Um desafio, você consegue encontrar as entidades, os atributos de cada entidade e o atributo identificador (chave) neste cenário? Utilize SOMENTE as informações que estão no cenário, não invente ou pressuponha conforme seus conhecimentos da realidade. Seja fiel ao cenário.

Uma companhia aérea está buscando desenvolver um sistema de gestão para melhorar a eficiência e a precisão em suas operações. A empresa deseja que você projete um Diagrama de Entidade-Relacionamento (DER) para representar as principais entidades do sistema. Considere as seguintes informações: Cada voo possui um número de voo único, uma rota específica (origem e destino), e uma data e horário de partida. Cada passageiro tem um número de identificação único, nome, endereço e informações de contato. Os passageiros podem fazer reservas em voos específicos. Cada reserva está associada a um passageiro. Cada reserva tem um número e deve incluir informações sobre a data da reserva e o status da reserva (confirmada, pendente, cancelada).

Resposta do Exercício 1

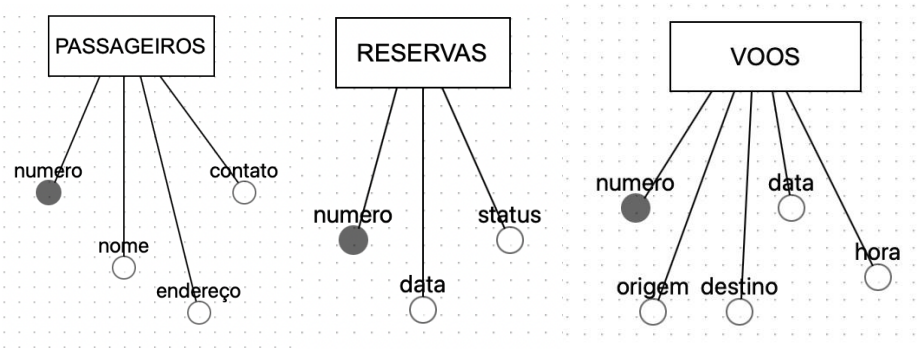


Figura 2.4: Entidades do cenário

- **Atributo identificador**

O atributo identificador é um campo único ou uma combinação de campos que define exclusivamente um registro.

Exemplo:

- **Campo único:** CPF do cliente
- **Combinação de Campos:** código do cliente, código do livro e data de locação

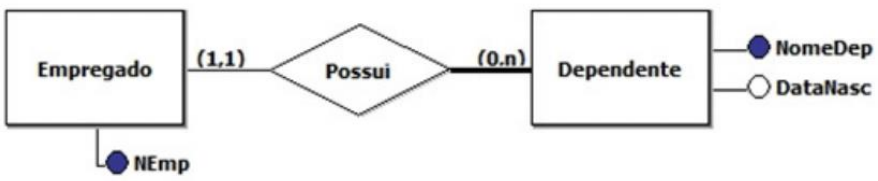


Figura 2.5: Exemplo de atributo identificador

- **Atributo multivalorado**

Um atributo multivalorado é um tipo de atributo que pode conter múltiplos valores para uma única instância de uma entidade. Isso significa que, ao contrário de um atributo simples, que contém apenas um valor para cada instância da entidade, um atributo multivalorado pode conter vários valores.

Por exemplo, considere uma entidade "Livro" em um banco de dados de uma biblioteca. O atributo "Autores" pode ser multivalorado, pois um livro pode ter mais de um autor. Portanto, para cada livro, pode haver múltiplos valores associados ao atributo "Autores", como "Autor1", "Autor2" etc.

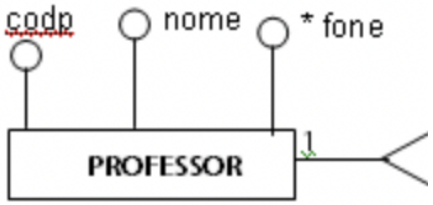


Figura 2.6: Exemplo de atributo multivalorado

- **Atributos compostos**

É um tipo de atributo que pode ser dividido em partes menores ou componentes. Em outras palavras, um atributo composto é formado por múltiplos subatributos que têm significado por si só, e que juntos compõem o atributo maior.

Por exemplo, considere uma entidade "Endereço" em um banco de dados de uma empresa. O atributo "Endereço" pode ser composto por vários subatributos, como "Rua", "Número", "Cidade", "Estado" e "CEP".

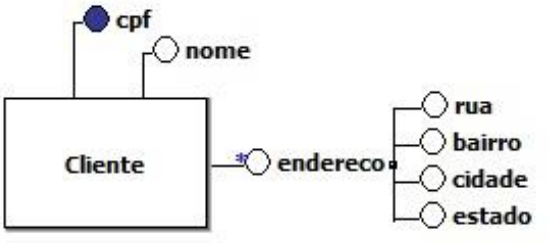


Figura 2.7: Exemplo de atributo composto

Exercício 2: VAMOS PRATICAR!

Encontre as entidades e atributos dos cenários seguintes. Defina atributos conforme seus conhecimentos da área.

1) Um sistema de reservas de voo permite que os passageiros busquem voos disponíveis, façam reservas e gerenciem suas viagens.

2) Um sistema de registro acadêmico de uma universidade gerencia informações sobre alunos, cursos e as disciplinas cursadas.

3) Um aplicativo de entrega de comida permite que os usuários peçam comida de restaurantes locais e rastreiem suas entregas.

4) Um sistema de gerenciamento de hospital mantém registros de pacientes, médicos, leitos e tratamentos médicos.

5) Um sistema de gerenciamento de estoque de uma loja controla o inventário de produtos, fornecedores e pedidos.

Resposta do Exercício 2

1) Um sistema de reservas de voo permite que os passageiros busquem voos disponíveis, façam reservas e gerenciem suas viagens.

Entidades: Passageiro, Voo, Reserva

Atributos:

Passageiro: Nome, Número de Identificação, Contato

Voo: Número do Voo, Origem, Destino, Data e Hora, Duração

Reserva: Número de Reserva, Assentos, Status

2) Um sistema de registro acadêmico de uma universidade gerencia informações sobre alunos, cursos e as disciplinas cursadas.

Entidades: Aluno, Curso, Disciplina, Professor, Matrícula

Atributos:

Aluno: Nome, Matrícula, Curso, E-mail

Curso: Nome, Código, Departamento

Disciplina: Nome, Código, Créditos

Professor: Nome, Departamento, Especialização

Matrícula: Ano, Período, Notas

3) Um aplicativo de entrega de comida permite que os usuários peçam comida de restaurantes locais e rastreiem suas entregas.

Entidades: Cliente, Restaurante, Pedido, Entregador, Produto

Atributos:

Cliente: Nome, Endereço, Método de Pagamento

Restaurante: Nome, Tipo de Cozinha, Horário de Funcionamento

Pedido: Número do Pedido, Itens, Status

Entregador: Nome, Veículo, Rota

Produto: Nome, Descrição, Preço

4) Um sistema de gerenciamento de hospital mantém registros de pacientes, médicos, leitos e tratamentos médicos.

Entidades: Paciente, Médico, Leito, Tratamento, Departamento

Atributos:

Paciente: Nome, Número de Identificação, Data de Nascimento

Médico: Nome, Especialidade, Horário de Atendimento

Leito: Número, Tipo, Status

Tratamento: Descrição, Data, Resultado

Departamento: Nome, Localização, Responsável

5) Um sistema de gerenciamento de estoque de uma loja controla o inventário de produtos, fornecedores e pedidos.

Entidades: Produto, Fornecedor, Pedido, Estoque, Categoria

Atributos:

Produto: Nome, Código, Preço, Quantidade em Estoque

Fornecedor: Nome, Contato, Endereço

Pedido: Número do Pedido, Data, Status

Estoque: Localização, Quantidade Mínima, Quantidade Máxima

Categoria: Nome, Descrição

- **Relacionamento**

Ampliando o exemplo da empresa, suponha que a empresa tenha anotado os dados das vendas feitas por cada vendedor a um determinado cliente e pretende armazenar essas

informações no banco de dados.

Vendas			
1	Vendedor Tomé Mascarenhas	José da Silva	R$ 6789,78
2	Vendedor André Costa	José da Silva	R$ 5000,00
3	Vendedor André Costa	Thiago Figueiredo	R$ 17000,00

Figura 2.8: Dados das vendas

Olhando as anotações das empresas, vemos que foram anotados os nomes dos vendedores e dos clientes, mas esses são atributos que já pertencem a outras entidades, as entidades de VENDEDORES e CLIENTES respetivamente, a VENDA possui como atributo próprio apenas o seu identificador e o valor. Isso nos indica que existe um relacionamento entre VENDA e VENDEDORES, e entre VENDA e CLIENTES.

Relacionamento são as associações que fazemos entre as entidades. Um vendedor pode fazer mais de uma venda. Observe na Figura 2.9 que o vendedor André Costa fez duas vendas, uma de R$5000,00 e outra de R$ 17000,00. O outro relacionamento é que um cliente pode ser o titular de várias vendas, o cliente José da Silva é um exemplo.

No DER representamos os relacionamentos por losangos ligados às entidades que se relacionam. Perceba que os atributos de VENDAS são apenas o seu ID (identificador) e o valor da venda. A relação é mostrada nos losangos. Não se preocupe com os dados neste momento. No modelo conceitual são identificados apenas os dados que manteremos no banco, quais são os atributos e se existe relação entre as entidades. Veja a representação gráfica no DER desses relacionamentos. Por enquanto, vamos ignorar esse pares (1,1), (0,n) e (1,1).

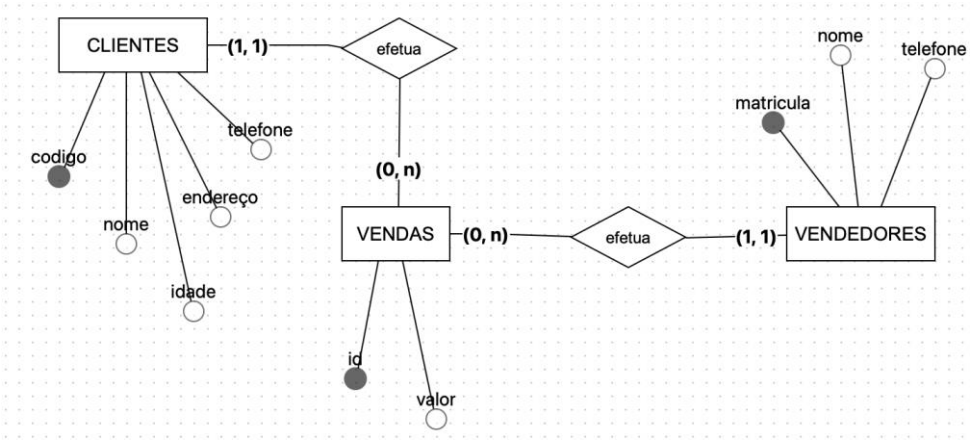


Figura 2.9: Modelo conceitual

No exemplo da companhia aérea, temos VOOS, PASSAGEIROS e RESERVAS, existe relação? Claro, um passageiro pode fazer várias reservas. Cada reserva poderia ter vários passageiros associados a ela, além de que uma reserva tem um ou mais voos registrados e cada voo pode aparecer em diversas reservas diferentes. Esse cenário de relacionamentos poderia ser descrito conforme a Figura 2.10.

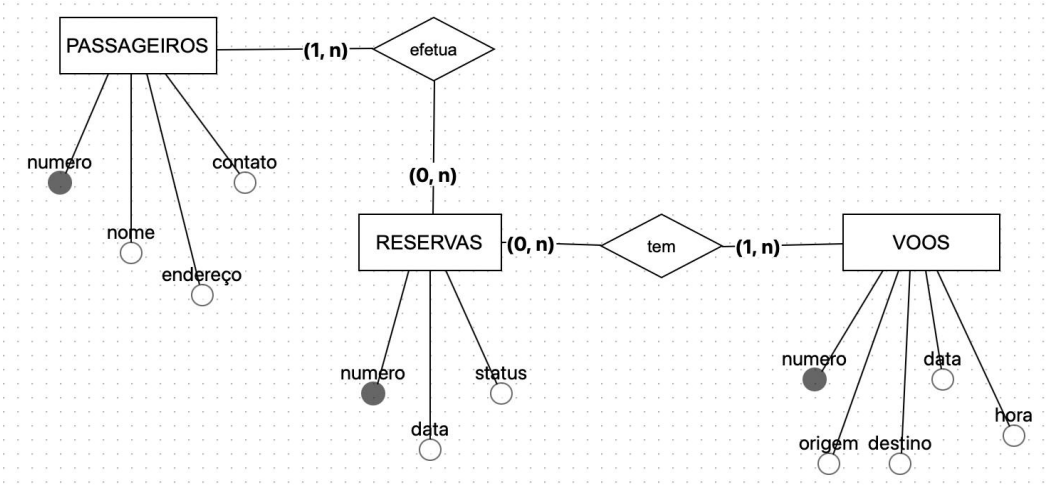


Figura 2.10: Modelo conceitual

2.2 Cardinalidade entre Relacionamentos

- **Cardinalidade**

Lembre-se, estamos fazendo um modelo conceitual abstrato, que não mostra os dados reais, apenas demostra o que deve ser registrado no banco de dados e como esses dados se

relacionam. Através deste modelo, devemos ser capazes de entender o que e como o banco de dados precisa ser projetado. Temos que colocar no diagrama o máximo de informação possível que dê as dicas do que precisa ser feito.

Já mostramos como descrever as entidades e seus atributos, também como identificar se as entidades se relacionam (se "conhecem"). Precisamos agora, estabelecer a cardinalidade do relacionamento, que são esses pares ordenados que aparecem nas linha: (1,1), (0,n) e (1,n).

A cardinalidade é o número mínimo e máximo de ocorrências de entidades associadas a outras. O primeiro valor do par ordenado é o mínimo e o segundo o máximo (mínimo, máximo):

- (1,1): no mínimo um e no máximo um
- (0,n) : no mínimo zero e no máximo muitos
- (1,n) : no mínimo um e no máximo muitos
- (n,n): no mínimo muitos e no máximo muitos

Os valores máximo e mínimo, representados por n, poderiam ser trocados por outros valores fixos, por exemplo (1,2) no mínimo um e no máximo dois. Essa escolha depende da organização que mantêm os dados. É preciso perguntar para o dono do dado.

Os valores 0 e 1 na cardinalidade mínima é entendida também como obrigatoriedade, se tiver valor 1 indica que o relacionamento deve associar pelo menos uma ocorrência, já o valor 0 diz que essa associação é opcional.

No exemplo de vendas, o vendedor André Costa pode fazer quantas vendas? Ele pode fazer nenhuma ou muitas, não é verdade? Então, no banco de dados podemos ter para um vendedor específico com nenhuma ou várias vendas atribuídas a ele. Modelamos isso colocando (0,n) no relacionamento entre as entidades VENDEDOR e VENDAS nesse sentido, e perto de VENDAS. Agora, se olharmos para uma única venda específica, por exemplo, a venda de número 2, quantos vendedores efetuaram essa venda? Apenas o vendedor André. Assim, modelamos o outro sentido da relação, colocando (1,1) do lado de VENDEDOR para dizer que para uma venda específica associamos apenas um vendedor.

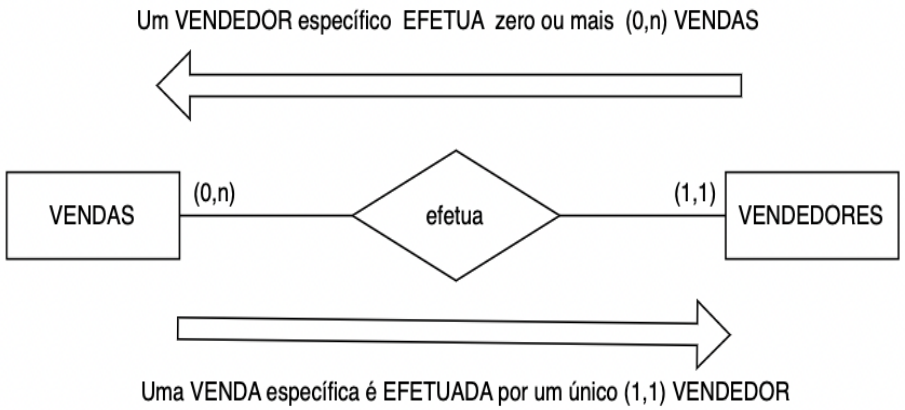


Figura 2.11: Sentido de leitura do relacionamento e cardinalidade entre entidades

Exercício 3: VAMOS PRATICAR!

Uma floricultura deseja informatizar suas operações. Inicialmente, deseja manter um cadastro de todos os seus clientes, mantendo informações como: RG, nome, telefone e endereço. Deseja também manter um cadastro contendo informações sobre os produtos que vende, tais como: nome do produto, tipo (flor, vaso, planta), preço e quantidade em estoque. Quando um cliente faz uma compra, a mesma é armazenada, mantendo informação sobre o cliente que fez o pedido, a data da compra, o valor total e os produtos comprados.

Monte o diagrama entidade-relacionamento.

Resposta do exercício 3

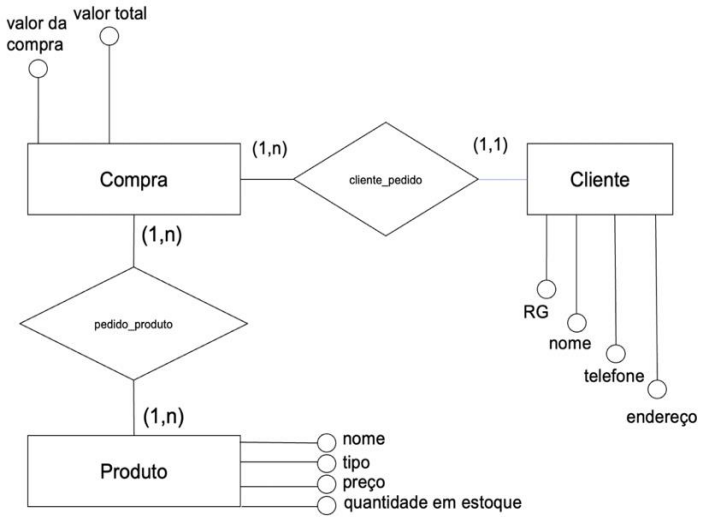


Figura 2.12: Modelo conceitual

- **Atributo de relacionamento**

Os atributos de relacionamento podem ser utilizados para expressar as restrições ou as características específicas do relacionamento. Por exemplo, se houver um relacionamento "Emprestado para", podemos ter atributos como "Data de empréstimo" e "Data de devolução" para indicar quando um LIVRO foi emprestado e quando ele precisa ser devolvido.

Perceba que os atributo "Data de empréstimo" e "Data de devolução" só existem se o relacionamento empréstimo for efetivado. Não é a mesma coisa que os atributos título do LIVRO ou matrícula do USUARIO que são independentes de um relacionamento.

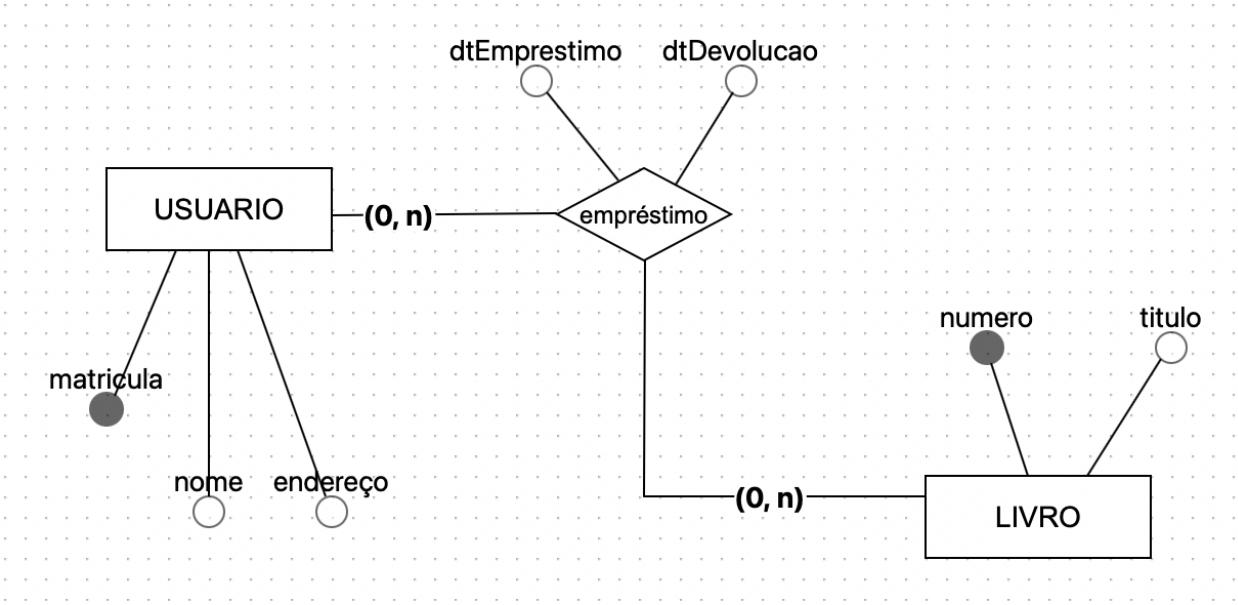


Figura 2.13: Modelo conceitual

Exercício 4: VAMOS PRATICAR!

Uma escola está desenvolvendo um sistema de gestão e precisa armazenar os dados de cada aluno: número de matrícula único, nome e endereço. Quer registrar também as informações da matrícula do aluno como data da matrícula e o valor da mensalidade. É importante registrar os cursos da escola que possuem número e nome. Um aluno ao se matricular escolhe o curso que ele pretende fazer. Nessa escola o aluno pode até se inscrever em diversos cursos, mas cada matrícula deve ser para apenas um dos cursos. O desafio é encontrar as entidades, atributos e estabelecer as relações com a cardinalidade correta entre as entidades definidas, considerando a matrícula como ponto de ligação entre os alunos e os cursos.

Monte o diagrama entidade-relacionamento.

Resposta do exercício 4

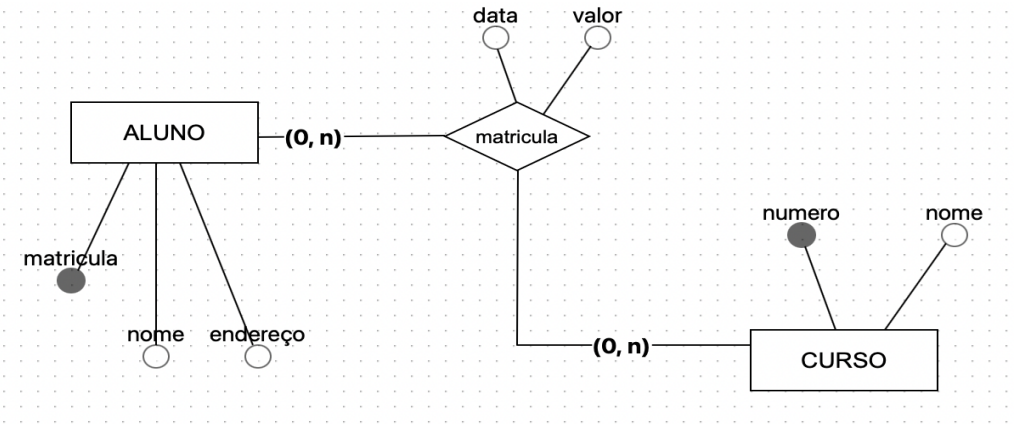


Figura 2.14: Modelo conceitual

Observe que a data da matrícula e o valor da mensalidade só existem quando o ALUNO se matricula no CURSO.

- **Relacionamento binário e ternário**

Relacionamento binário é o que acontece entre duas entidades, vimos isso nos exemplos anteriores. Mas existe também o relacionamento ternário, entre três entidades. Podemos ter também relacionamentos quaternários, etc., dependendo da quantidade de tabelas que se relacionam. Vamos focar apenas nos ternários.

Vamos usar o exemplo do livro Projeto de Banco de Dados do Carlos Alberto Heuser para exemplificar uma situação ternária. O modelo representa CIDADE, DISTRIBUIDOR e PRODUTO. O distribuidor pode distribuir produtos em diversas cidades. O modelo proposto está na Figura 2.15, perceba que o losango do relacionamento ternário liga as três entidades. Mas como ler essas cardinalidades? A cardinalidade neste caso é vista aos pares únicos de entidades.

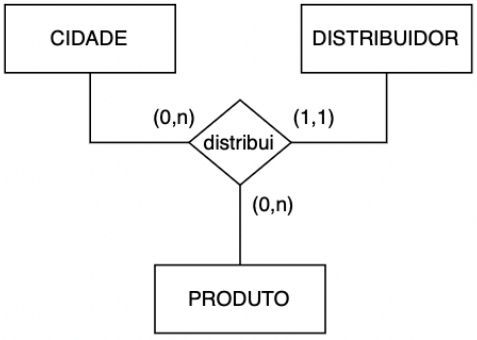


Figura 2.15: Sentido de leitura do relacionamento e cardinalidade entre entidades

Na Figura 2.16, tem a análise de CIDADE e PRODUTO. Imagine uma cidade qualquer (Goiânia) e um produto vendido (Feijão). O que o modelo diz a respeito da quantidade de distribuidores desse único produto para essa única cidade? Olhamos para a cardinalidade próxima a DISTRIBUIDOR, o modelo diz que uma CIDADE tem para cada PRODUTO um e somente um DISTRIBUIDOR, ou seja, não teremos um produto sendo distribuído por mais de um distribuído. Existe exclusividade produto x distribuidor em cada cidade.

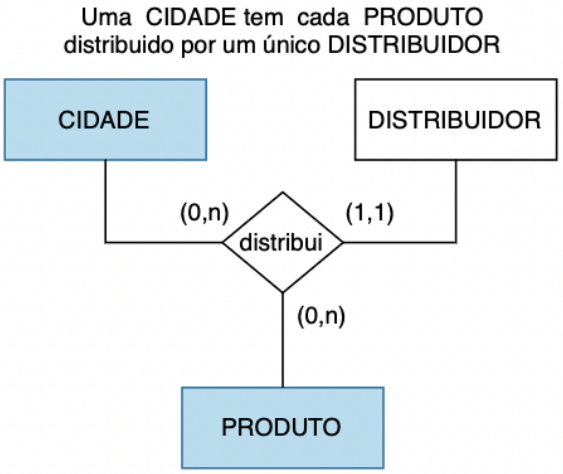


Figura 2.16: Sentido de leitura PRODUTO, CIDADE => DISTRIBUIDOR

Olhando o sentido DISTRIBUIDOR, PRODUTO em relação à CIDADE na Figura 2.17. Cada distribuidor distribui cada produto dele em várias cidades.

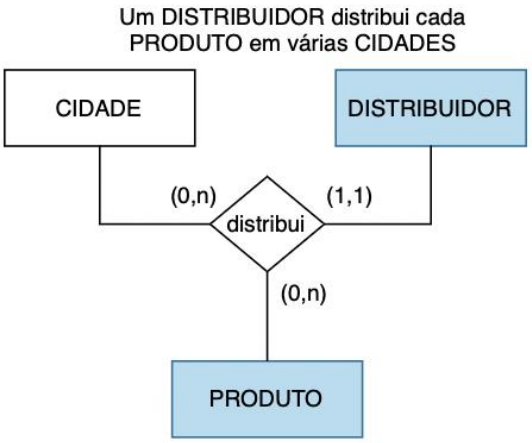


Figura 2.17: Sentido de leitura DISTRIBUIDOR, PRODUTO => CIDADE

Por último, o sentido DISTRIBUIDOR, CIDADE em relação ao PRODUTO na Figura 2.18. Cada distribuidor distribui em cada cidade vários produtos.

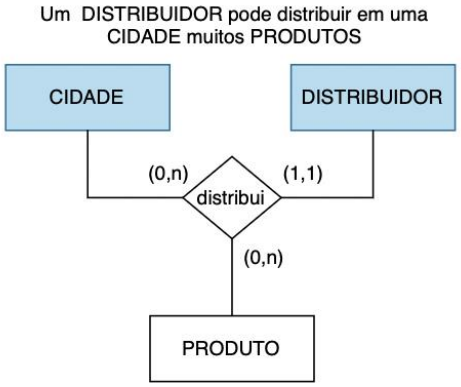


Figura 2.18: Sentido de leitura DISTRIBUIDOR, CIDADE => PRODUTO

2.3 Modelo lógico

Uma vez que os dados já foram mapeados na forma de um DER é preciso definir o tipo de SGBD (relacional, rede, orientado a objetos, etc.) que será usado e transformar o modelo conceitual em modelo lógico. O modelo lógico depende do tipo de SGBD que iremos usar detalhando como as informações serão armazenadas internamente.

Neste curso, abordaremos o modelo relacional no qual os dados são armazenados em formato de tabela.

Considere o modelo conceitual da figura 2.19

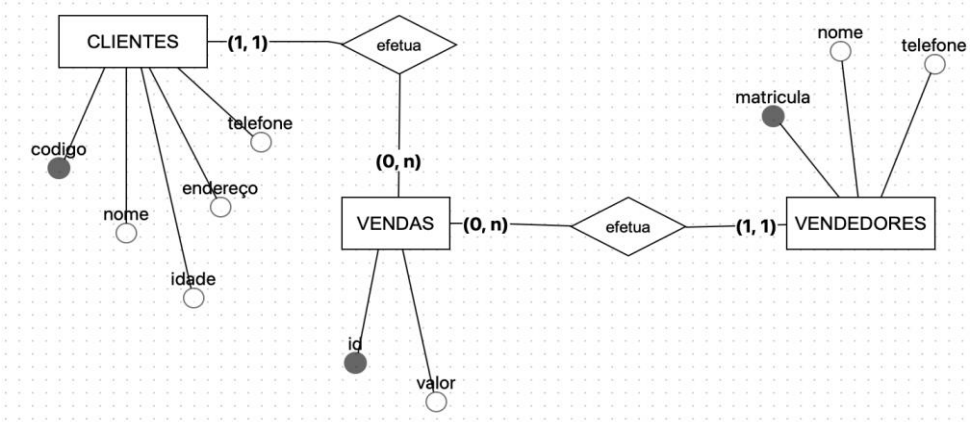


Figura 2.19: Modelo conceitual

Um modelo lógico tem originem em um modelo conceitual. Enquanto o modelo

conceitual é uma representação de alto nível que descreve os conceitos e as relações entre eles em um domínio específico, se concentrando na semântica e nos relacionamentos entre entidades, ignorando detalhes de implementação. O modelo lógico é uma representação mais detalhada e estruturada dos dados, capturando como os dados serão armazenados e organizados no sistema de gerenciamento de banco de dados (SGBD). Ele traduz os conceitos abstratos do modelo conceitual em estruturas de dados concretas, como tabelas, colunas e chaves.

Para o exemplo da Figura 2.19 o modelo lógico teria uma estrutura como a seguinte:

- **CLIENTES (codigo, nome, idade, endereco, telefone);**
- **VENDEDORES (matricula, nome, telefone);**
- **VENDAS (id, codigo_cliente, mat_vendedor, valor):**
  - codigo_cliente **referencia CLIENTES;**
  - mat_vendedor **referencia VENDEDORES.**

Sendo que:

- CLIENTES:
  - Tabela: CLIENTES;
  - Atributos: código (chave primária), nome, idade, endereco, telefone.

- VENDEDORES:
  - Tabela: VENDEDORES;
  - Atributos: matricula (chave primária), nome, telefone.

- VENDAS:
  - Tabela: VENDAS
  - Atributos: id (chave primária), valor
  - **Chaves estrangeiras:**
    - **codigo_cliente referenciando a tabela CLIENTES (indicando qual cliente fez a compra)**
    - **mat_vendedor referenciando a tabela VENDEDORES (indicando qual vendedor realizou a venda)**

Importante notar que uma chave estrangeira (ou chave externa) é um conceito em bancos de dados relacionais que estabelece uma relação entre duas tabelas. Ela é uma coluna ou conjunto de colunas em uma tabela que faz referência à chave primária ou a uma chave única em outra tabela. Essa referência cria um vínculo entre as duas tabelas, permitindo que os dados em uma tabela estejam relacionados aos dados em outra tabela.

Quando uma chave estrangeira é definida em uma tabela, ela garante que os valores na coluna de chave estrangeira só possam aparecer se já existirem na coluna correspondente na tabela referenciada (a tabela pai). Isso ajuda a manter a integridade dos dados, evitando que sejam inseridos valores inválidos nas colunas de referência.

Por exemplo, considerando o modelo lógico fornecido anteriormente:

Na tabela VENDAS, as colunas **codigo_cliente** e **mat_vendedor** são chaves estrangeiras.

codigo_cliente faz referência à chave primária (código) na tabela CLIENTES.

mat_vendedor faz referência à chave primária (matricula) na tabela VENDEDORES.

Isso significa que cada entrada na tabela VENDAS deve corresponder a um cliente e a um vendedor que já existe nas tabelas CLIENTES e VENDEDORES, respectivamente. Se um registro de venda tentar referenciar um cliente ou vendedor que não existe nas tabelas já existente, isso violaria a integridade referencial e seria impedido pelo banco de dados.

Ao invés de colocar o nome dos vendedores e cliente colocamos as matrículas e códigos.

VENDAS

ID	MAT_VENDEDOR	CODIGO_CLIENTE	VALOR
1	2335	234	6789,78
2	3426	234	5000,00
3	3426	785	17000,00

Figura 2.20: Tabela de VENDAS com chaves estrangeiras mat_vendedor e código_cliente

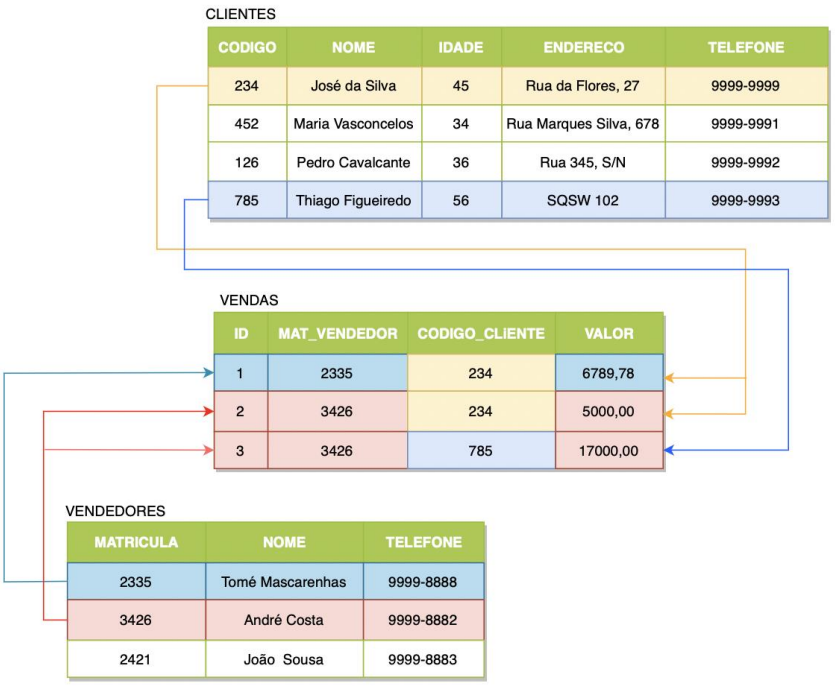


Figura 2.21: Relacionamento entre as tabelas (visão com dados)

UNIDADE 3

NORMALIZAÇÃO DE TABELAS

Nesta unidade, exploraremos os princípios fundamentais da normalização de tabelas em bancos de dados. A normalização é um processo crucial para projetar esquemas de banco de dados que garantam a integridade dos dados, a eficiência das consultas e a facilidade de manutenção do sistema. Ao compreender os conceitos de dependência funcional e as formas normais, os profissionais de banco de dados podem criar estruturas robustas e otimizadas para armazenar e manipular informações.

Ao final dos estudos, você deverá ser capaz de: entender os conceitos de dependência funcional e as estratégias de normalização de um modelo de dados.

3.1 Dependência Funcional

A dependência funcional é um conceito da matemática que estabelece uma relação entre os atributos de dois conjuntos, indicando que valores de um conjunto influenciam ou determinam os valores de outro dentro de uma relação.

Considere que não existe repetição de nome em um conjunto que armazena o nome de funcionários. Observe que os valores do conjunto MATRICULA, identificam um elemento no conjunto NOME. Se pegarmos o nome André Costa, a matrícula dele unicamente só pode ser 3426, não existe mais de uma opção

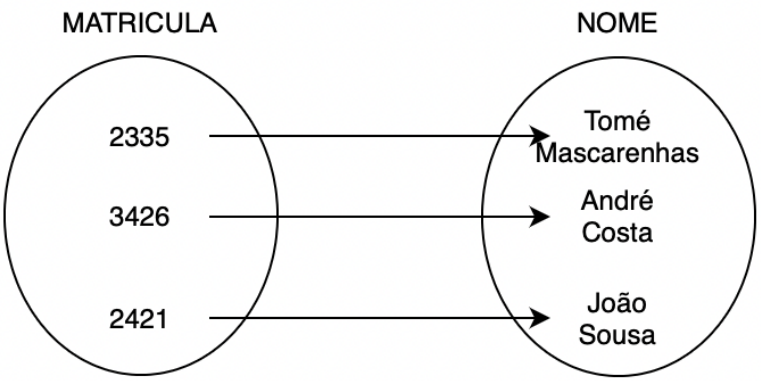


Figura 3.1: Dependência funcional

Podemos ver que cada matrícula está relacionada a um único nome:

2335 → {Tomé Mascarenhas}

3426 → {André Costas}

2421 → {João Sousa}

Agora, observe os conjuntos MATRICULA e DEPARTAMENTO, com uma matrícula podendo ter mais de um departamento associado a ela, como é o caso da matrícula 2335. Assim, existe dependência funcional entre MATRICULA e NOME, mas não existe entre MATRICULA e DEPARTAMENTO.

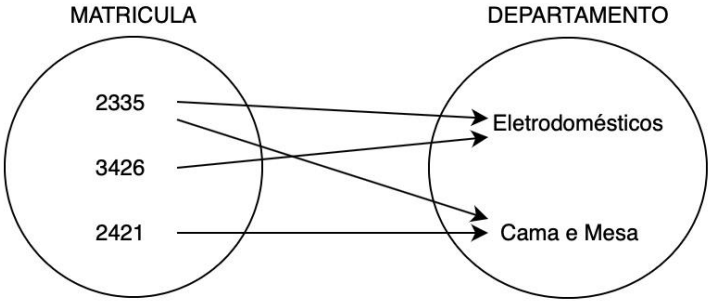


Figura 3.2: Não tem dependência funcional

Observa as relações entre os valores concretos:

2335 → {Eletrodoméstico, Cama e Mesa}

3426 → {Eletrodomésticos}

2421 → {Cama e Mesa}

Resumimos assim, a dependência funcional:

Dados dois conjuntos A e B, diz-se que B é funcionalmente dependente de A, ou A determina B, ou B depende de A, se cada valor de A estiver associado a um e somente um valor de B. Pode ser representado assim:

A → B

Esse conceito é usado em banco de dados. Segundo ELMASRI e NAVATHE (2019), "uma dependência informal é uma restrição entre dois conjuntos de atributos de um banco de dados". Falando assim, fica complicado entender, vamos exemplificar.

VENDEDORES

MATRICULA	NOME	TELEFONE	DEPARTAMENTO
2335	Tomé Mascarenhas	9999-8888	Eletrodomésticos
3426	André Costa	9999-8882	Eletrodomésticos
2421	João Sousa	9999-8883	Cama e Mesa
2335	Tomé Mascarenhas	9999-8888	Cama e Mesa

Figura 3.3: Tabela vendedores

Vamos transformar o exemplo dos conjuntos em tabelas. Considere a Tabela VENDEDORES, e os dados registrados como na Figura 3.3, se o atributo MATRICULA determina de forma única o NOME do vendedor, então, temos a dependência funcional, que é o caso do exemplo. Representamos a dependência com a seta, como a seguir

Matricula → Nome

Isso significa que o NOME depende da MATRICULA do vendedor, existe uma relação um-para-um. Agora, considere a dependência:

Matricula → Departamento

Com base nos dados, esta dependência está errada. Não existe uma dependência funcional clara entre MATRICULA e DEPARTAMENTO, pois uma MATRICULA pode pertencer a diferentes DEPARTAMENTOS ao longo do tempo. Observe que o vendedor de matrícula 2335 aparece em dois departamentos: Eletrodomésticos e Cama e Mesa.

A relação inversa também não existe.

Departamento → Matricula

Isso porque um departamento determina mais de uma matrícula.

Eletrodomésticos → {2334, 3426}

Cama e Mesa → {2121, 2335}

A dependência funcional em banco de dados pode ser verificada entre vários atributos. Por exemplo, ser do tipo A → BC, ou AB → C.

Vamos exemplificar com MATRICULA, DEPARTAMENTO e HORARIO. Pelo exemplo da tabela da Figura 3.4, existe dependência.

VENDEDORES

MATRICULA	NOME	TELEFONE	DEPARTAMENTO	HORARIO
2335	Tomé Mascarenhas	9999-8888	Eletrodomésticos	MATUTINO
3426	André Costa	9999-8882	Eletrodomésticos	INTEGRAL
2421	João Sousa	9999-8883	Cama e Mesa	INTEGRAL
2335	Tomé Mascarenhas	9999-8888	Cama e Mesa	VESPERTNO

Figura 3.4: Tabela vendedores

São exemplos de dependências que poderíamos analisar:

Matricula → Departamento, Horário

Departamento, Horário → Matricula

Você consegue indicar se existe ou não dependência funcional para esses exemplos, considerando os dados da tabela representada na Figura 3.4?

Para ficar mais fácil pensar na possível resposta, vamos colocar os dados:

Matricula → Departamento, Horário

2335 → {(Eletrodoméstico, Matutino), (Cama e Mesa, Vespertino)}

3426 → {(Eletrodoméstico, Integral)}

2421 → {(Cama e Mesa, Integral)}

Neste caso, não existe dependência funcional.

Departamento, Horário → Matricula

(Eletrodoméstico, Matutino), → 2335

(Cama e Mesa, Vespertino) → 2335

(Eletrodoméstico, Integral) → 3426

(Cama e Mesa, Integral) → 2421

Neste caso, existe dependência funcional porque a matrícula 2335 tem relação com mais de um par Departamento, Horário.

Regras das dependências funcionais

- **Separação/Decomposição**

Se A → BC

então:

A → B

A → C

Um exemplo desta relação para os dados da tabela da Figura 3.4:

Se Matricula → Nome, Telefone

então:

Matricula → Nome

Matricula → Telefone

- **Acumulação**

Se A → B

então:

AC → B

Um exemplo desta relação para os dados da tabela da Figura 3.4:

Se Matricula → Nome

então:

Matricula, Horario → Nome

- **Transitividade**

Se A → B e B → C

então:

A → C

Um exemplo desta relação para os dados da tabela da Figura 3.4:

Se Matricula → Nome e Nome → Telefone

então:

Matricula → Telefone

- **Decomposição**

Se A → BC e BC → D

então:

AC → D

Para este exemplo, a nossa tabela não possui opção.

Exercício 1: VAMOS PRATICAR!

Um desafio, para a estrutura de tabela da Figura 3.4, agora, com os atributos Matrícula, Nome, Telefone, Endereço, Departamento, Horário, Salário. Considere que as regras para os dados que serão armazenados sejam os seguintes:

- A matrícula é única por vendedor e chave primária na tabela.
- Podem existir vendedores homônimos (com o mesmo nome).
- O telefone é único por vendedor.
- Pode ter dois vendedores morando em um mesmo endereço.
- Um vendedor só pode estar lotado em um departamento.
- Um vendedor só trabalha em um único horário por departamento.
- Os salários podem ser diferentes entre vendedores, mas um vendedor só tem um salário.

Para uma tabela com estas regras, diga se existe ou não dependência funcional para os exemplos abaixo:

- Matricula → Departamento, Horário
- Matricula → Endereco
- Endereco → Matricula
- Matricula → Telefone, Endereco
- Departamento, Horário → Matricula
- Matricula → Departamento, Horário
- Matricula → Salário
- Salário → Matricula
- Salário → Matricula, Departamento

3.2 Introdução à Normalização

A determinação da estrutura de uma tabela é muito importante. Vamos olhar um exemplo que temos usado nesta unidade.

VENDEDORES

MATRICULA	NOME	TELEFONE	DEPARTAMENTO	HORARIO
2335	Tomé Mascarenhas	9999-8888	Eletrodomésticos	MATUTINO
3426	André Costa	9999-8882	Eletrodomésticos	INTEGRAL
2421	João Sousa	9999-8883	Cama e Mesa	INTEGRAL
2335	Tomé Mascarenhas	9999-8888	Cama e Mesa	VESPERTNO

Figura 3.5: Tabela vendedores

Esta tabela tem um problema. Imagine que o nome de um dos departamentos seja alterado, teríamos que mudar todas as linhas que possuam esse departamento. Esse é um problema causado pela redundância. O nome do departamento se repete em vários lugares. O ideal é que estivesse em apenas um local e mudando em um, todos estariam atualizados.

Um outro problema claro é o fato de neste exemplo um vendedor poder estar lotado em mais de um DEPARTAMENTO, esse é um atributo chamado multivalorado. A solução da tabela foi repetir o registro da matrícula do 2335, além de consumir mais espaço, isso é uma redundância que poderia causar inconsistência pois se algum dado desse vendedor se alterar, tem que lembrar de mudar nos dois pontos.

A normalização é o processo que ajuda a resolver problemas como esses através da simplificação da estrutura do modelo de dados, determinando uma estrutura melhor. A normalização é a decomposição de uma tabela com o objetivo de eliminar redundâncias e inconsistências, reduzir falhas e facilitar as manutenções.

3.2.1 Primeira Forma Normal (1FN)

Uma tabela está na 1ª Forma Normal se não possui atributos multivalorados ou compostos. Vamos lembrar o que são atributos multivalorados e compostos (ou não atômicos).

Atributos compostos são atributos que podem ser subdivididos em vários atributos. Por exemplo, a tabela:

Funcionario (Codigo, Nome, Endereco)

O atributo endereço é composto, uma vez que ele pode ser dividido em vários atributos como Rua, Número, Complemento, Cidade, Estado, Cep.

Um atributo multivalorado é indicado como uma lista ou conjunto. Por exemplo, o

atributo telefone, caso uma pessoa possa ter mais de um telefone.

Transformação para a 1FN

- Separar o atributo composto em seus componentes.
- Para o atributo multivalorado retirá-lo da tabela criando uma nova que tem o mesmo conjunto de atributos chave, mais o atributo multivalorado também como chave.

VENDEDORES

MATRICULA	NOME	TELEFONE	ENDERECO	DEPARTAMENTO	HORARIO
2335	Tomé Mascarenhas	9999-8888	Rua 25, n. 35, Goiânia, 74.999-000	[Eletrodomésticos; Cama e Mesa]	MATUTINO
3426	André Costa	9999-8882	Rua das Flores, n. 234, Goiânia, 74,673-000	Eletrodomésticos	INTEGRAL
2421	João Sousa	9999-8883	Rua Camilo Capixaba, S/N, Goiânia, 74.999-000	Cama e Mesa	INTEGRAL

Figura 3.6: Tabela vendedores

VENDEDORES (Matricula, Nome, Telefone, Endereco (Rua, Cidade, CEP), Departamento, Horario).

Analisando, temos o atributo DEPARTAMENTO multivalorado, um vendedor com dois departamentos de lotação, e o ENDERECO como multivalorado contendo os campos (Rua, Cidade, CEP).

Para aplicar a primeira forma normal a estrutura ficaria:

- Separar o atributo composto em seus componentes.

VENDEDORES (Matricula, Nome, Telefone, Rua, Cidade, CEP, Departamento, Horario).

Para o atributo multivalorado retirá-lo da tabela criando uma nova que tem o mesmo conjunto de atributos chave, mais o atributo multivalorado também como chave.

VENDEDORES (Matricula, Nome, Telefone, Rua, Cidade, CEP, Horario)

DEPARTAMENTO (Matricula, Departamento)

Uma outra opção para substituir o atributo multivalorado é criar atributos no número máximo de valores estabelecido para o grupo, essa é uma abordagem menos genérica e que pode introduzir muitos valores nulos.

VENDEDORES (Matricula, Nome, Telefone, Rua, Cidade, CEP, Departamento, Horario)

Poderíamos resolver abaixo e não criar a segunda tabela.

VENDEDORES (Matricula, Nome, Telefone, Rua, Cidade, CEP, Departamento1, Departamento2, Horario)

Não vamos usar essa opção, observe que aqueles vendedores que não possuíssem dois departamentos teriam essa informação como nula. Um outro problema é se surgisse a possibilidade de alocar o vendedor a três ou mais departamentos, teríamos que mudar a estrutura do modelo.

REGRA: uma relação está na 1FN se:

- Todo os atributos devem ser atômicos e monovalorados.
- A entidade não deve conter grupos multivalorados.

3.2.2 Segunda Forma Normal (2FN)

Para estar na 2ª Forma Normal todo atributo não chave deve ser totalmente dependente da chave primária e não apenas de parte dela.

Transformação para a 2FN

Separar os atributos que dependem de um subconjunto da chave, decompondo a relação em duas ou mais relações.

Vamos usar como exemplo a relação:

PEDIDO (Numero, Data, Codigo_peca, Descricao_peca, Quantidade, Preco)

Verificar a dependência funcional. Pela regra, todos os atributos não chave devem depender de Numero e Codigo_peca juntos e não apenas de um deles.

A Data depende de Número, não depende de Codigo_peca.

Numero → Data

A Descricao_peca só depende de Codigo_peca, não depende de Número.

Codigo_peca → Descricao_peca

A Quantidade e Preço dependem sim da chave completa. Número sozinho ou Codigo_peca não dependem os dois atributos.

Numero , Codigo_peca → Quantidade, Preco.

O que faremos para resolver o problema é dividir o conjunto de atributos que compõem a chave primária em subconjuntos. Para cada um desses subconjuntos, criaremos uma relação, tendo esse subconjunto como sua chave primária. No nosso exemplo, teremos

uma relação com Numero e outra com Codigo_peca. Incluir os atributos da relação original em uma das relações criadas junto com a chave mínima da qual ele depende e atribuir um nome a cada relação resultante.

- **PEDIDO (Numero, Data)**
- **PRODUTO (Codigo_peca, Descricao_peca)**
- **VENDA (Numero, Codigo_peca, Quantidade, Preco)**

REGRA: uma relação está na 2FN se:

- Se estiver na 1FN.
- Todos os atributos que não pertencem à chave dependem de toda a chave (e não de um subconjunto da chave).

3.2.3 Terceira Forma Normal (3FN)

Nesta forma, normalmente não existem atributos não chave que sejam dependentes de outros atributos não chave, ou seja, possuem dependência transitiva.

Se A → B e B → C

então:

A → C

Transformação para a 3FN

- Separar os atributos que dependem de outro atributo que não pertença a chave, decompondo a relação em duas ou mais relações.

Vamos usar como exemplo a relação:

CLIENTE (cpf, nome, telefone, cod_vendedor, nome_vendedor)

Verificar a dependência funcional, pela regra não existem atributos não chave que sejam dependentes de outros atributos não chave.

O nome_vendedor depende do cod_vendedor, ambos são não chave.

cod_vendedor → nome_vendedor

O que vamos fazer é retirar cod_vendedor, que não é uma chave candidata, da relação e, junto, os atributos que dependem deste determinante. Criaremos uma nova relação contendo todos os atributos da relação original que dependem deste determinante. O determinante será a chave primária da nova relação.

- **CLIENTE (cpf, nome, telefone)**
- **VENDEDOR (cod_vendedor, nome_vendedor)**

REGRA: uma relação está na 3FN se:

- Se estiver na 2FN.
- Todos os atributos que não pertencem à chave dependem de nenhum atributo que também não pertence à chave.

UNIDADE 4

MODELO FÍSICO

Nesta unidade, vamos nos aprofundar no entendimento e na aplicação prática de conceitos essenciais em banco de dados, começando pelo Modelo Físico. Este tópico nos permitirá compreender como os dados são realmente armazenados e organizados em um ambiente de banco de dados.

Em seguida, exploraremos a Linguagem SQL, uma ferramenta fundamental para interagir com bancos de dados relacionais. Através dela, você será capaz de realizar uma variedade de operações, desde consultas simples até manipulações complexas de dados, proporcionando maior flexibilidade e controle sobre suas bases de dados.

Ao final dos estudos, você deverá ser capaz de: criar modelos físicos e consultas básicas usando Linguagem SQL.

4.1 Modelo Físico

Na etapa final do processo de desenvolvimento de um banco de dados, o modelo físico emerge como a representação tangível e concreta do projeto. Este modelo é o resultado de um esforço contínuo de ajustes e refinamentos, refletindo as necessidades e as demandas dinâmicas do ambiente operacional. É importante ressaltar que o modelo físico está sujeito a mudanças ao longo do ciclo de vida do banco de dados, adaptando-se às evoluções e às exigências do sistema em uso.

4.2 Linguagem SQL

A Linguagem SQL, ou Structured Query Language, é uma ferramenta essencial no mundo da gestão de bancos de dados. Utilizada para definir a estrutura das tabelas que irão armazenar os dados, bem como para estabelecer os relacionamentos entre esses dados. A SQL desempenha um papel crucial na organização e na gestão eficiente das informações.

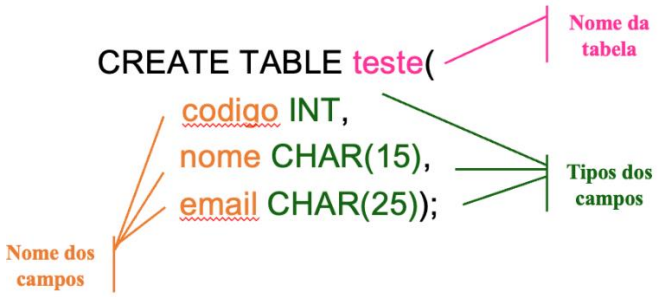
Além disso, a SQL permite extrair insights valiosos por meio de consultas complexas e análises detalhadas.

4.3 Criação de Banco de Dados e Tabelas

4.3.1. Definição de Tabelas, Atributos e Chaves

A criação de um banco de dados começa com a definição das tabelas que irão compor sua estrutura fundamental. Cada tabela é projetada com atributos específicos que descrevem as características dos dados que serão armazenados. Além disso, são estabelecidas chaves primárias e estrangeiras para garantir a integridade e a consistência dos dados, facilitando a manipulação e a recuperação das informações armazenadas.

Em SQL a cláusula para criar uma tabela é o CREATE TABLE. Sendo informado no mínimo os atributos da tabela e o tipo.



Campo	Tipo	Descrição
codigo	Integer	Código do funcionário(não nulo)
nome	Char(40)	Nome do funcionário (não nulo)
setor	Char(2)	Setor onde o funcionário trabalha
cargo	Char(20)	cargo do funcionário
salario	Decimal(10,2)	salário do funcionário
Chave Primária		Será o campo código

Figura 4.1: Tabela empregador

A cláusula SQL para a figura 4.1 seria:

```
CREATE TABLE Empresa.Empregados (
    codigo INT NOT NULL,
    nome VARCHAR(40) NOT NULL,
```

setor **VARCHAR(2) NOT NULL,**

cargo **VARCHAR(20) NOT NULL,**

salario **REAL NOT NULL,**

PRIMARY KEY (codigo));

A parte **PRIMARY KEY** (codigo) define que o atributo código será a chave primária.

Mais informações sobre SQL, consultar a bibliografia DAMAS, Luís. **SQL - Structured Query Language**. LTC, 2005.

4.4 (DQL) Consultas Em SGBD

4.4.1 Introdução às Consultas SQL

As consultas SQL desempenham um papel central nos sistemas de gerenciamento de banco de dados (SGBD), permitindo que os usuários recuperem, atualizem e manipulem dados de maneira eficiente e intuitiva. Nesta seção, exploraremos os conceitos fundamentais por trás das consultas SQL, incluindo a seleção de dados de tabelas, a filtragem de resultados e a ordenação de informações.

Supondo que para a tabela da figura 4.1 se deseje realizar as seguintes consultas:

- Apresentar a listagem completa dos registros da tabela Empregados.
- Apresentar uma listagem dos nomes e dos cargos de todos os registros da tabela Empregados.
- Apresentar uma listagem dos nomes dos empregados do setor 1.
- Listagem dos nomes e dos salários por ordem de nome (a-z).
- Listagem dos nomes e dos salários por ordem de nome em formato descendente (z-a).

A resposta seria:

- Apresentar a listagem completa dos registros da tabela Empregados:

```
SELECT * FROM Empresa.Empregados.
```

- Apresentar uma listagem dos nomes e dos cargos de todos os registros da tabela Empregados:

SELECT nome, cargo FROM Empresa.Empregados.

- Apresentar uma listagem dos nomes dos empregados do setor 1:

SELECT nome FROM Empresa.Empregados WHERE setor = '1'.

- Listagem dos nomes e dos salários por ordem de nome (a-z):

SELECT nome, salario FROM Empresa.Empregados ORDER BY nome DESC.

- Listagem dos nomes e dos salários por ordem de nome em formato descendente (z-a):

SELECT nome, salario FROM Empresa.Empregados ORDER BY nome ASC.

Mais informações sobre SQL, consultar a bibliografia DAMAS, Luís. **SQL - Structured Query Language**. LTC, 2005.

UNIDADE 5

MANIPULAÇÃO DE DADOS

Nesta unidade, vamos aplicar todos os conhecimentos adquiridos nas unidades anteriores em um projeto prático completo.

Ao final dos estudos, você deverá ser capaz de: resolver um problema com a criação do modelo conceitual, lógico e físico de banco de dados.

5.1 Projeto Prático

Considere o cenário.

Os clientes podem desejar encontrar os filmes estrelados pelo seu ator predileto. Por isso, é necessário manter a informação dos atores de cada filme. Para cada ator os clientes desejam saber o nome real e/ou nome artístico, bem como a data de nascimento. O serviço de streaming possui muitos clientes cadastrados. Somente clientes cadastrados podem alugar filmes. Sobre os filmes, cada um tem um ID e um título. Além disso, um filme pode ter apenas uma categoria, por exemplo, comédia. Cada categoria tem um código e uma descrição. Para cada cliente é necessário saber seu CPF, nome e seu sobrenome, seu telefone e seu endereço. Além disso, cada cliente recebe um número de associado. Finalmente, desejamos saber que filmes cada cliente tem alugado. Um cliente pode ter vários filmes em um instante no tempo. Deve ser registrado a data dos aluguéis.

As tabelas abaixo, apresentam os dados que devem ser armazenados no banco de dados.

Cliente

cpf	nome	sobrenome	endereco	telefone
1	Joao	Silva	Rua A	86753412
1	Joao	Silva	Rua A	89237772
2	Maria	Silva	Rua B	78658922
3	José	Silva	Rua C	86893377
4	Pedro	Silva	Rua D	97678277

Ator

id	data_de_nascimento	nome_popular	nome_artistico
1	1987-10-17	Brian Cool	NULL
2	1978-04-23	Alan McDonald	MC B
3	2000-09-25	Deborah Hilton	NULL

Categoria

codigo	descricao
1	Comédia
2	Ação

Filme

id	titulo	categoria
1	Filme 1	Comédia
2	Filme 2	Comédia
3	Filme 3	Acão

Locacao

cliente	filme	dtLocacao
Joao	Filme 1	2014-03-07
Maria	Filme 1	2014-03-07
José	Filme 2	2014-03-07

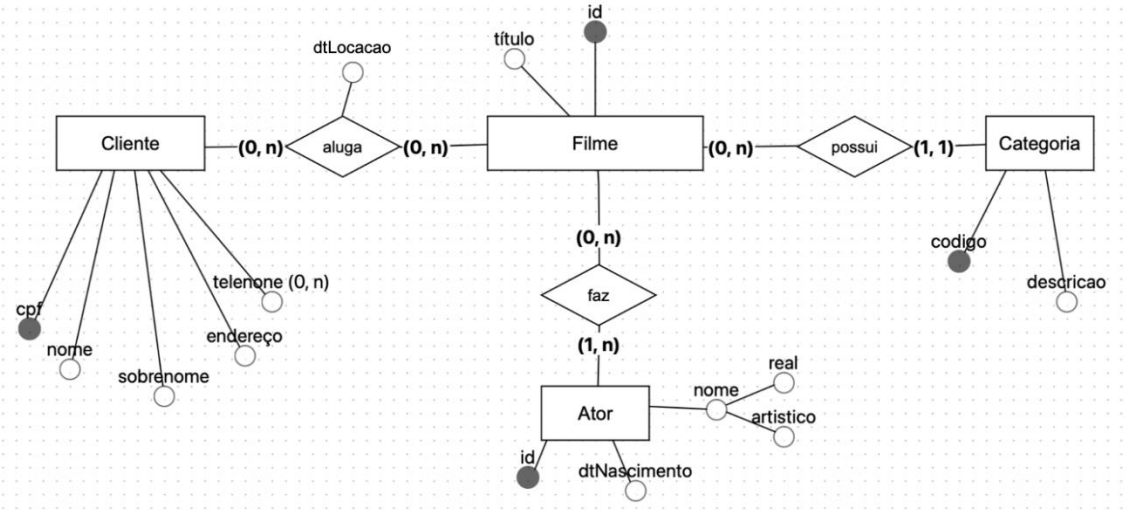
Filmes que os atores atuaram

ator	filme
Alan McDonald	Filme 1
Brian Cool	Filme 2
Alan McDonald	Filme 2
Alan McDonald	Filme 3
Deborah Hilton	Filme 3

Crie o modelo conceitual, o modelo lógico (NORMALIZADO) e o modelo físico. Monte as SQL para responder:

- Apresentar a listagem completa dos registros dos Clientes.
- Apresentar uma listagem dos nomes de todos os Atores.
- Apresentar uma listagem dos filmes locados do filme 1.
- Apresentar uma listagem dos filmes e a descrição da categoria.

1) Modelo Conceitual



2) Modelo Lógico

- Cliente (**cpf**, nome, sobrenome, endereço)
- Telefone (**fk_cpf_cliente**, **telefone**)
  - fk_cpf_cliente referencia Cliente
- Filme (**id**, título, fk_cod_categoria)
  - fk_cod_categoria referencia Categoria
- Categoria (**codigo**, descricao)
- Locacao (**fk_cpf_cliente**, **fk_id_filme**, **dtLocacao**)
  - fk_cpf_cliente referencia Cliente
  - fk_id_filme referencia Filme
- Ator (**id**, data_de_nascimento,nome_popular, nome_artistico)
- Filme_Atores (**fk_id_filme**, **fk_id_ator**)
  - fk_id_fime referencia Filme
  - fk_id_ator referencia Ator

Modelo Físico

CREATE SCHEMA Streaming.

CREATE TABLE Streaming.Cliente (

cpf **VARCHAR(11) NOT NULL,**

```
                nome VARCHAR(40) NOT NULL,
                sobrenome VARCHAR(40) NOT NULL,
                endereco VARCHAR(40) NOT NULL,
                PRIMARY KEY (cpf));

CREATE TABLE Streaming.Telefone (
                fk_cpf_cliente VARCHAR(11) NOT NULL,
                telefone VARCHAR(10) NOT NULL,
                PRIMARY KEY (fk_cpf_cliente, telefone));

CREATE TABLE Streaming.Filme (
                id INT NOT NULL,
                titulo VARCHAR(80) NOT NULL,
                 fk_cod_categoria INT NOT NULL,
                PRIMARY KEY (id));

CREATE TABLE Streaming.Categoria (
                codigo INT NOT NULL,
                descricao VARCHAR(40) NOT NULL,
                PRIMARY KEY (codigo));

CREATE TABLE Streaming.Locacao (
                fk_cpf_cliente VARCHAR(11) NOT NULL,
                fk_id_filme INT NOT NULL,
                dtLocacao VARCHAR(10) NOT NULL,
                PRIMARY KEY (fk_cpf_cliente, fk_id_filme, dtLocacao));

CREATE TABLE Streaming.Ator (
                id INT NOT NULL,
                data_de_nascimento VARCHAR(10) NOT NULL,
                nome_popular VARCHAR(40) NOT NULL,
                nome_artistico VARCHAR(40) ,
```

```
                    PRIMARY KEY (id));

CREATE TABLE Streaming.Filme_Ator (
                    fk_id_filme INT NOT NULL,
                    fk_id_ator INT NOT NULL,
                    PRIMARY KEY (fk_id_filme, fk_id_ator));

ALTER TABLE Streaming.Telefone ADD CONSTRAINT fk_cpf_cliente
   FOREIGN KEY (fk_cpf_cliente )
   REFERENCES Cliente (cpf)
   ON DELETE RESTRICT
   ON UPDATE NO ACTION;

ALTER TABLE Streaming.Filme ADD CONSTRAINT fk_cod_categoria
   FOREIGN KEY (fk_cod_categoria)
   REFERENCES Categoria (codigo)
   ON DELETE RESTRICT
   ON UPDATE NO ACTION;

ALTER TABLE Streaming.Locacao ADD CONSTRAINT fk_cpf_cliente_locacao
   FOREIGN KEY (fk_cpf_cliente)
   REFERENCES Cliente (cpf)
   ON DELETE RESTRICT
   ON UPDATE NO ACTION;

ALTER TABLE Streaming.Locacao ADD CONSTRAINT fk_id_filme
   FOREIGN KEY (fk_id_filme )
   REFERENCES Filme (id)
   ON DELETE RESTRICT
   ON UPDATE NO ACTION;

ALTER TABLE Streaming.Filme_Atores ADD CONSTRAINT fk_id_filme_atores
```

```
    FOREIGN KEY (fk_id_filme)
    REFERENCES Filme (id)
    ON DELETE RESTRICT
    ON UPDATE NO ACTION;

ALTER TABLE Streaming.Filme_Ator ADD CONSTRAINT fk_id_ator
    FOREIGN KEY (fk_id_ator )
    REFERENCES Ator (id)
    ON DELETE RESTRICT
    ON UPDATE NO ACTION;
```

3) Inserção dos dados

```
INSERT INTO Streaming.Cliente(cpf, nome, sobrenome, endereco) VALUES ('1', 'Joao', 'Silva', 'Rua A');
INSERT INTO Streaming.Cliente(cpf, nome, sobrenome, endereco) VALUES ('2', 'Maria', 'Silva', 'Rua B');
INSERT INTO Streaming.Cliente(cpf, nome, sobrenome, endereco) VALUES ('3', 'José', 'Silva', 'Rua C');
INSERT INTO Streaming.Cliente(cpf, nome, sobrenome, endereco) VALUES ('4', 'Pedro', 'Silva', 'Rua D’);
INSERT INTO Streaming.Telefone(fk_cpf_cliente, telefone) VALUES ('1', '86753412');
INSERT INTO Streaming.Telefone(fk_cpf_cliente, telefone) VALUES ('1', '89237772');
INSERT INTO Streaming.Telefone(fk_cpf_cliente, telefone) VALUES ('2', '78658922');
INSERT INTO Streaming.Telefone(fk_cpf_cliente, telefone) VALUES ('3', '86893377');
INSERT INTO Streaming.Telefone(fk_cpf_cliente, telefone) VALUES ('4', '97678277’);
INSERT INTO Streaming.Categoria(codigo, descricao) VALUES (1, 'Comédia');
INSERT INTO Streaming.Categoria(codigo, descricao) VALUES (2, 'Ação’);
INSERT INTO Streaming.Filme(id, titulo, fk_cod_categoria) VALUES (1, 'Filme 1',1);
INSERT INTO Streaming.Filme(id, titulo, fk_cod_categoria) VALUES (2, 'Filme 2',1);
INSERT INTO Streaming.Filme(id, titulo, fk_cod_categoria) VALUES (3, 'Filme 3',2);
INSERT INTO Streaming.Locacao(fk_cpf_cliente, fk_id_filme, dtLocacao) VALUES (1, 1, '2014-03-07');
```

```
INSERT INTO Streaming.Locacao(fk_cpf_cliente, fk_id_filme, dtLocacao) VALUES (2, 1,
'2014-03-07');
INSERT INTO Streaming.Locacao(fk_cpf_cliente, fk_id_filme, dtLocacao) VALUES (3, 2,
'2014-03-07');
INSERT INTO Streaming.Ator(id, data_de_nascimento, nome_popular, nome_artistico)
VALUES (1,'1987-10-17', 'Brian Cool', NULL);
INSERT INTO Streaming.Ator(id, data_de_nascimento, nome_popular, nome_artistico)
VALUES (2,'1978-04-23', 'Alan McDonald', 'MC B');
INSERT INTO Streaming.Ator(id, data_de_nascimento, nome_popular, nome_artistico)
VALUES (3,'2000-09-25', 'Deborah Hilton', NULL);
INSERT INTO Streaming.Filme_Ator(fk_id_filme, fk_id_ator) VALUES (1,2);
INSERT INTO Streaming.Filme_Ator(fk_id_filme, fk_id_ator) VALUES (2,2);
INSERT INTO Streaming.Filme_Ator(fk_id_filme, fk_id_ator) VALUES (2,1);
INSERT INTO Streaming.Filme_Ator(fk_id_filme, fk_id_ator) VALUES (3,3);
INSERT INTO Streaming.Filme_Ator(fk_id_filme, fk_id_ator) VALUES (3,2);
INSERT INTO Streaming.Filme_Ator(fk_id_filme, fk_id_ator) VALUES (3,1);
```

4) Consultas

```
#Apresentar a listagem completa dos registros dos Clientes;
SELECT * FROM Streaming.Cliente;

#Apresentar uma listagem dos nomes de todos os Atores;
SELECT * FROM Streaming.Ator;

#Apresentar uma listagem dos dos filmes locados do filme 1;
SELECT * FROM Streaming.Locacao WHERE fk_id_filme = 1;

#Apresentar uma listagem dos filmes e a descricao da categoria;
SELECT f.id, f.titulo, c.descricao AS categoria
FROM Streaming.Filme AS f, Streaming.Categoria AS c
WHERE f.fk_cod_categoria = c.codigo ;
```

PARA FINALIZAR

Chegamos ao término da nossa jornada de exploração sobre Banco de Dados, e reconheço sua persistência e dedicação até aqui. Parabenizo-o por sua perseverança e comprometimento durante todo o curso. Espero que este material tenha sido uma fonte de enriquecimento para você, proporcionando valiosos conhecimentos sobre o intrigante universo dos bancos de dados.

Durante nossa jornada, exploramos os fundamentos essenciais de bancos de dados, desde a distinção entre dados e informações, até a importância dos Sistemas Gerenciadores de Banco de Dados (SGBD) na organização e manipulação eficiente das informações. Abordamos temas como o Modelo Entidade-Relacionamento (MER), a normalização de tabelas e os conceitos-chave da linguagem SQL.

Acredito que os bancos de dados são a espinha dorsal de qualquer sistema de informação moderno, e compreender profundamente seus princípios e práticas é fundamental para qualquer profissional da área de tecnologia da informação. Com o advento de grandes volumes de dados e a necessidade crescente de análise e processamento eficiente, o conhecimento em banco de dados torna-se ainda mais crucial.

Lembre-se de que os conceitos e habilidades adquiridos aqui não apenas enriquecerão sua carreira profissional, mas também podem ser aplicados em uma variedade de contextos pessoais e profissionais. A capacidade de projetar e gerenciar bancos de dados eficazes não apenas aumenta sua empregabilidade, mas também abre portas para oportunidades de inovação e solução de problemas em diversas áreas.

Agora, convido você a continuar sua jornada de aprendizado e aprofundamento neste campo emocionante. Explore novas técnicas, aprimore suas habilidades e mantenha-se atualizado com as últimas tendências e tecnologias em banco de dados. O campo de bancos de dados está em constante evolução, e seu potencial como profissional é verdadeiramente ilimitado.

Parabenizo você por sua dedicação e participação ativa neste curso. Desejo-lhe sucesso em sua trajetória profissional e espero que este material seja apenas o começo de uma jornada repleta de realizações e descobertas no vasto mundo dos bancos de dados.

Joelma de Moura Ferreira

Sobre o autor

Joelma de Moura Ferreira é doutora em Ciência da Computação pela Universidade Federal de Goiás, com mestrado em Ciência da Computação pela Universidade Federal de Goiás, MBA em Gerenciamento de Projetos pela Fundação Getúlio Vargas, especialização em Redes de Computadores pela Universidade Salgado de Oliveira, MBA em Tecnologia para Negócios: AI, Data Science e Big Data pela Pontifícia Universidade Católica do Rio Grande do Sul e graduação em Ciência da Computação pela Universidade Católica de Goiás. Atuou por mais de 20 anos como docente de graduação e pós-graduação em diversas instituições de ensino superior, incluindo Faculdade Sul-Americana, Universidade Paulista, Faculdade Estácio de Sá de Goiás, Pontifícia Universidade Católica, Centro Universitário Alves Farias. Desempenhou a função de coordenadora do curso de graduação de Sistemas de Informação e dos cursos de pós-graduação em Gestão de Projetos, Gestão de Tecnologia da Informação e Arquitetura e Engenharia de Software no Centro Universitário Alves Faria, onde também exerceu a atividade de pesquisadora no Mestrado em Desenvolvimento R Fora do domínio acadêmico, exerce a função de Cientista de Dados no Tribunal de Justiça do Distrito Federal e Territórios.

Referências Bibliográficas

DAMAS, L. **SQL - Structured Query Language.** Rio de Janeiro: LTC, 2005.

ELMASRI, R.; NAVATHE, S. B. **Sistemas de banco de dados.** 7ª ed. São Paulo: Pearson Prentice Hall, 2019.

GRAVES, M. **Projeto de Banco de Dados com XML.** São Paulo: Pearson Prentice Hall, 2003.

LEAL, G. C. L. **Linguagem, programação e banco de dados: guia prático de aprendizagem.** Curitiba: Intersaberes, 2015.

MACHADO, F. N. R. **Banco de Dados - Projeto e Implementação**. São Paulo: Erica, 2020.

MEDEIROS, L. F. **Banco de dados: princípios e prática.** Curitiba: Intersaberes, 2012.

PUGA, S.; FRANÇA, E.; GOYA, M. **Banco de dados: implementação em SQL, PL/SQL e Oracle 11g**. São Paulo: Pearson Prentice Hall, 2015.

VICCI, C. (org.). **Banco de Dados**. Rio de Janeiro: LTC, 2004.

IPOG