BANCO DE DADOS

AULA 1

Prof. Ricardo Sonaglio Albano e Profª. Silvie Guedes Albano

CONVERSA INICIAL

Fundamentos de Banco de Dados

O tema Banco de Dados é um conhecimento que deve ser adquirido pelos profissionais da área de tecnologia, pois basicamente todas as informações, de uma forma ou outra, encontram-se armazenadas em Bancos de Dados.

Neste estudo, abordaremos os conceitos e as técnicas que você precisará para ingressar no universo de Banco de Dados Relacional, abrangendo desde os aspectos de modelagem até a utilização da linguagem *Structured Query Language* (SQL) para a manipulação das informações.

Apresentaremos e discutiremos tópicos que o ajudarão a desenvolver habilidades para a interpretação e análise de informações, ingressando, assim, em um processo de reflexão sobre o desenvolvimento de um Banco de Dados e as melhores práticas de atuação como profissional da área.

Aliando a teoria e a prática, apoiaremos você em cada passo desse caminho, acompanhando-o no processo de construção de uma base de dados e discutindo sobre as melhores abordagens, metodologias existentes, suas formas de aplicação e implementação em soluções práticas no desenvolvimento.

Nesta primeira etapa, apresentaremos os principais conceitos sobre o Sistema Gerenciador de Banco de Dados (SGBD), os tipos de modelos de dados, as etapas da modelagem de dados, a construção do Modelo Entidade-Relacionamento (MER) e a influência da cardinalidade exercida sobre o modelo de dados.

Vale salientar que desenvolver uma forte base conceitual sobre Banco de Dados servirá de suporte fundamental na construção prática de projetos de Banco de Dados. São conceitos que você deverá manter em sua memória, pois com certeza os utilizará frequentemente no dia a dia de sua profissão.

Bons estudos e sucesso!

TEMA 1 – CONCEITOS, DEFINIÇÕES E MODELOS

Em todas as organizações existe a necessidade do armazenamento de informações. Além de uma forma adequada para definir o armazenamento dessas informações, os usuários realizam consultas em determinado conjunto de dados, atualizam ou modificam a estrutura dos dados e eliminam informações não necessárias.

Também é importante ressaltar que as empresas se baseiam nas informações para a tomada de decisões e que, por esse motivo, a informação tem grande importância e valor para as organizações.

Sendo assim, a informação, além de ser precisa, deve ser cada vez mais ágil e instantânea. Essa busca pela informação tem ajudado no desenvolvimento dos chamados *sistemas de processamento de informações*.

1.1 Dado x informação

É importante destacar a diferença entre *dado* e *informação*, pois no cotidiano é comum serem utilizados como sinônimos. Porém, quando se trata de computadores, é importante que tenhamos a diferença bem definida entre esses dois termos.

Para exemplificarmos essa diferença de forma simples, vamos imaginar uma simples fórmula matemática de adição. Ela é composta por **dados** que serão processados pelo computador. O resultado obtido é a **informação**.

Vejamos:

Figura 1 – Dado x informação

Fonte: Albano; Albano, 2022.

Perceba que a fórmula apresentada na imagem acima possui três dados: dois números e a operação a ser realizada (símbolo da adição).

À medida que os dados são organizados ou agrupados de uma forma significativa, tornam-se uma informação. Portanto, podemos afirmar que a informação é um conjunto de fatos organizados de tal forma que adquirem um valor adicional além do valor do dado em si. Dessa forma, devido ao seu grande valor, necessitam ser armazenadas de forma apropriada, permitindo o rápido acesso e, ao mesmo tempo, garantindo a integridade e segurança desses fatos organizados. Para isso, existe o Banco de Dados que trataremos logo a seguir.

1.2 Banco de Dados (BD)

Existem diversos conceitos utilizados para definir um Banco de Dados. Vejamos alguns deles:

- Coleção de dados persistentes usados pelos sistemas de aplicação de determinada empresa;
- Conjunto de dados inter-relacionados representando informações de um domínio específico;
- Sistema que registra e mantém dados baseados em computador;
- Sistema computadorizado de armazenamento de registros, cujo objetivo é armazenar informações e permitir aos usuários buscar e atualizar essas informações quando necessário;
- Todo o conjunto de dados que segue determinado modelo de dados.

Resumindo, um Banco de Dados (BD) é o local onde são armazenados os dados de uma aplicação.

Exemplo de um Banco de Dados: sistema de delivery, em que a empresa deseja armazenar as informações sobre os clientes, produtos, entregadores, restaurantes, entre outros. Cada um desses elementos serão as entidades básicas sobre as quais a aplicação de delivery precisará registrar informações.

1.2.1 Modelos de Banco de Dados

É a representação do Banco de Dados, demonstrando a relação existente entre os dados.

Os modelos de Banco de Dados são:

- Hierárquico;
- Rede;
- Orientado a objetos;
- Relacional;
- NoSQL ou não-relacional.

1.2.1.1 Modelo hierárquico

O Sistema de Banco de Dados Hierárquico (*Hierarquical Database System* – HDS) surgiu na década de 1960 com a primeira linguagem de Banco

de Dados, conhecida como DL/I, desenvolvida pela IBM e a North American Aviation.

Nesse modelo os dados são armazenados em estruturas no formato de árvore. Cada estrutura de dados parte de um nodo raiz (nó) e se ramifica criando relações pai-filho com outras classes de dados, gerando relações de um para muitos elementos.

A desvantagem está na rigidez da estrutura de dados, em que, no caso de alterações da classe principal ou de classes dependentes, obrigaria que fosse refeito todo o Banco de Dados.

1.2.1.2 Modelo de rede

Também conhecido como Sistema de Banco de Dados em Rede (*Network Database System* – NDS), esse modelo surgiu entre as décadas de 1960 e 1970 como uma extensão do modelo hierárquico, adicionando recursos para criação de mais de uma relação pai-filho e estabelecendo relações entre os seus elementos.

Esse modelo possui a vantagem de uma pesquisa mais rápida e flexível, pois não depende de um único nó raiz como vetor de inicialização da pesquisa. Apesar disso, o modelo de rede ainda apresenta os problemas relatados no modelo anterior no que se refere à relação ao projeto de estrutura do modelo hierárquico. Qualquer alteração feita em uma classe de dados implicaria na criação de uma nova estrutura para suportar àquela alteração.

Nesse caso, as relações entre os registros são feitas por meio de ponteiros que armazenam endereços de memória, indicando os endereços em que os dados realmente se encontram armazenados.

O modelo em rede trabalha com registros vinculados uns aos outros, gerando conjuntos de dados.

1.2.1.3 Modelo orientado a objetos

Incorpora as características da metodologia de desenvolvimento orientado a objetos com os conceitos de Sistema Gerenciador de Banco de Dados (SGBD).

Nesse modelo de dados, a informação é organizada e, consequentemente, armazenada sob a forma de objetos, inclusive com a

definição do comportamento de cada um, cuja manipulação é feita por meio de métodos. Não apresenta a obrigatoriedade referente às limitações quanto aos tipos de dados característicos de um SGBD e as linguagens de consulta.

1.2.1.4 Modelo relacional

O Modelo de Dados Relacional (*Relational Data Model* – RDM) foi criado na década de 1970 por um pesquisador da IBM, Dr. Edgar Frank Codd, sendo descrito no artigo "*Relational Model of Data for Large Shared Data Banks*". Tal modelo é, inclusive, considerado o primeiro modelo de dados descrito teoricamente e tinha por objetivo desenvolver uma representação mais simples dos dados, por meio de um modelo matemático de conjuntos de tabelas inter-relacionadas.

Esse modelo era bastante inovador, pois deixava de lado todos os conceitos anteriores e apresentava uma visão totalmente nova com estruturas mais flexíveis, tanto na forma de representação das relações entre os dados como também no processo de manutenção da estrutura.

O Banco de Dados relacional está estruturado em relações (tabelas) relacionadas entre si, relações essas que possuem atributos. Tais conceitos serão estudados em um próximo momento.

Vale salientar que, nesse estudo, adotaremos o Banco de Dados relacional.

1.2.1.5 Modelo NoSQL ou modelo não-relacional

Atualmente, além do modelo de Banco de Dados relacional, existem também os Bancos de Dados denominados NoSQL, que armazenam e manipulam as informações de uma forma diferente, sendo utilizados de forma mais flexível no acesso e manipulação de grandes volumes de dados e em situações em que a informação é apresentada de forma não estruturada ou semiestruturada.

Alguns exemplos de aplicação desse tipo de modelo são: redes sociais, *streaming*, *games*, *Internet of Things* (IoT), *Big Data*, entre outros.

Esse tipo de modelo apresenta as seguintes características:

- Flexibilidade: caracteriza-se pela ausência de esquemas padronizados de definição de estrutura de dados, permitindo que os dados sejam armazenados de forma mais livre, podendo manipular, de forma mais

simples, qualquer formato de dados. Dessa forma, possibilita-se não só um desenvolvimento mais rápido, mas também uma melhor performance no tratamento de volumes muito grandes de requisições, gerando respostas mais rápidas;

- Escalabilidade: são projetados para expandir seus recursos de forma horizontal, isto é, adicionando máquinas comuns (*nodes*) a estrutura já disponível, bem mais baratas que os caros servidores. Consequentemente, expande-se sua capacidade para suportar o aumento de tráfego, sua disponibilidade no recebimento de requisições e aumenta a performance em tempo de execução e resposta;
- Alta performance: focado na otimização de modelos de dados e padrões de acesso, gerando maior performance em comparação a outros modelos de dados.

No NoSQL existem quatro tipos principais de modelos de dados: chave-valor, documento, gráfico e família de colunas.

Vale salientar que existem situações em que sua aplicação não é considerada a mais adequada e, dessa forma, o Banco de Dados relacional é a melhor alternativa.

TEMA 2 – SISTEMA GERENCIADOR DE BANCO DE DADOS (SGBD) E APLICAÇÕES DE BANCO DE DADOS

Entre os dados armazenados fisicamente no disco rígido e os usuários do sistema, que manipulam esses dados, existe uma camada de software conhecida como Sistema Gerenciador de Banco de Dados (SGBD).

O Sistema Gerenciador de Bancos de Dados é a interface entre os dados de baixo nível, armazenados em um Banco de Dados, e os usuários e as aplicações, que desejam acessar e manipular esses dados.

O sistema de Banco de Dados é um sistema de manutenção de registros por computador e, para tanto, possui diversos recursos à disposição do usuário, possibilitando a realização de várias operações, como por exemplo:

- Adição, remoção e atualização de tabelas no Banco de Dados;
- Inserção de novos dados;
- Recuperação, atualização e exclusão de dados.

Como objetivos, o SGBD deve:

- Ocultar dos usuários os detalhes mais técnicos de funcionamento do Banco de Dados, também conhecido como *abstração de dados*;
- Prover independência de dados às aplicações (estrutura física de armazenamento e a estratégia de acesso).

Destacamos alguns SGBDs amplamente utilizados: DB2 (IBM), Microsoft SQL Server, Oracle, MySQL, PostgreSQL, entre outros.

2.1 Características de um SGBD

Um SGBD deve ter algumas características que são essenciais para o seu funcionamento, tais como:

- Controle de redundância: consiste no armazenamento de uma mesma informação em locais diferentes, provocando duplicidade e possíveis inconsistências. Em um Banco de Dados, as informações encontram-se armazenadas em um único local, não existindo duplicação dos dados;
- Compartilhamento dos dados e concorrência: o SGBD deve incluir o controle de concorrência no acesso aos dados, possibilitando o compartilhamento dos dados e garantindo que, se vários usuários realizarem operações de atualização sobre um mesmo conjunto de dados, o resultado dessas operações será correto;
- Controle de acesso: o SGDB deve disponibilizar recursos para o gerenciamento de acesso dos usuários, definindo permissões, como, por exemplo, um usuário poderá ter acesso total, enquanto outro usuário terá apenas acesso de leitura de dados;
- Controle de transações: uma transação é um conjunto de operações sobre o Banco de Dados que devem ser executadas integralmente e sem falhas ou interrupções;
- Possibilidade de múltiplas interfaces: um SGBD pode ser acessado por diversas interfaces diferentes (aplicação WEB, aplicativo mobile, entre outros). Logo, o SGBD deve garantir que todos tenham acesso aos dados da mesma forma;
- Restrições de integridade: deve garantir que as regras de integridades definidas no momento da criação do Banco de Dados sejam aplicadas

corretamente. Por exemplo, a data de nascimento não pode ser maior que a data atual;

- Backup e recuperação de dados: garantir o backup e a restauração de dados é tarefa essencial para qualquer SGBD, mesmo que as falhas não sejam originadas pelo próprio SGBD, mas sim falhas de software ou hardware;
- Independência de dados: os dados armazenados no Banco de Dados devem ser totalmente independentes de qualquer aplicação que o acesse. Inclusive, os aplicativos não possuem a descrição real de como os dados estão fisicamente armazenados;
- Eliminação de inconsistências: trata-se do armazenamento da informação em um único local com acesso descentralizado, sendo compartilhado por vários sistemas, mantendo uma informação confiável. A inconsistência ocorre quando um mesmo campo tem valores diferentes em sistemas diferentes;
- Padronização dos dados: permite que os atributos sejam definidos baseando-se em um formato de armazenamento predeterminado. Por exemplo, os atributos de data armazenados no formato “aaaa/mm/dd”.

2.2 Pontos de atenção em um SGBD

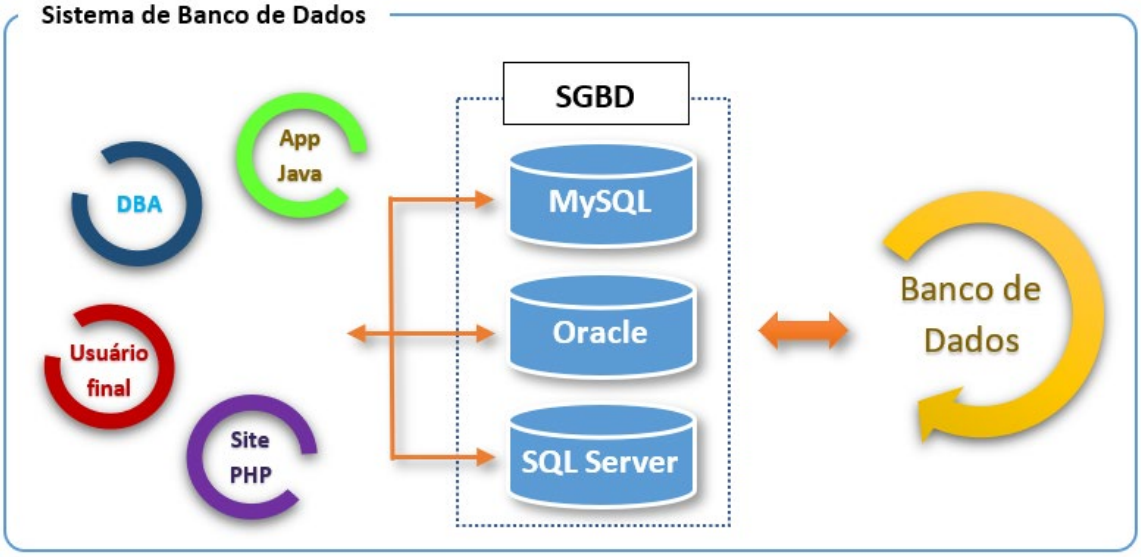
- Sem dispositivos de controle adequados, a segurança pode estar em risco. Por exemplo, no caso de acesso não autorizado aos dados;
- A integridade das informações pode ser comprometida se não houver mecanismos de controle. Por exemplo, no caso de manipulação concorrente de dados;
- Levantamento de dados de forma muito precisa e cuidadosa para evitar que informações não correspondam à realidade;
- A administração do sistema de Banco de Dados pode se tornar muito complexa em ambientes distribuídos, com grande volume de informações manipuladas por uma grande quantidade de usuários.

2.3 Sistema de Banco de Dados

Um sistema de Banco de Dados é um ecossistema que engloba quatro componentes principais:

1. Hardwares: dispositivos em que as informações são armazenadas fisicamente;
2. SGBD: software que permite o acesso dos usuários às informações armazenadas no Banco de Dados;
3. Aplicações: softwares que realizam requisições para o SGBD com a finalidade de interagir com as informações do Banco de Dados;
4. Usuários: estão divididos em três categorias:
    - Programador de aplicações – responsável pela definição dos programas de aplicação que utilizam o Banco de Dados;
    - Administrador de Banco de Dados ou DBA – responsável pelo controle e gerenciamento;
    - Usuário final – interage com o sistema a partir de um computador conectado ao Banco de Dados.

Figura 2 – Representação de um sistema de Banco de Dados



Fonte: Albano; Albano, 2022.

2.4 Administrador de Banco de Dados (DBA)

Podendo ser um ou mais responsáveis pela tarefa de gerenciamento do Banco de Dados, esses profissionais são responsáveis pela autorização de acesso ao Banco de Dados para os demais usuários, coordenação e monitoração do uso, ou seja, são eles que coordenam todas as atividades do sistema de Banco de Dados e possuem conhecimento sobre os recursos de informação da empresa e suas necessidades. Suas funções incluem:

- Definição do Banco de Dados;
- Estrutura de armazenamento e definição de acesso aos dados;
- Esquema físico e organização dos dados;
- Conceder acesso aos usuários;
- Cuidar da integridade dos dados;
- Acompanhar o desempenho do Banco de Dados;
- Atividades de manutenção e backups.

TEMA 3 – MODELAGEM DE DADOS

O objetivo da modelagem de dados é a representação do cenário observado e suas necessidades, analisando, documentando, normalizando e detectando como as informações relacionam-se entre si, bem como fornecendo processos de validação que nos permitem avaliar a veracidade dos modelos desenvolvidos.

Um Banco de Dados pode ser descrito ou modelado em vários níveis de abstração, sendo, assim, definidos três modelos básicos:

1. Modelo conceitual;
2. Modelo lógico;
3. Modelo físico.

Antes de iniciar a construção de um Banco de Dados é essencial a existência do projeto desse banco. Os modelos ajudam a demonstrar graficamente o Banco de Dados que será construído. Um modelo de dados é a descrição formal de um Banco de Dados.

3.1 Projeto de Banco de Dados

Todo bom sistema de Banco de Dados deve apresentar um projeto com objetivo de organizar as informações e utilizar técnicas para promover a criação de um sistema que apresente boa performance, além de facilitar o processo de manutenção que porventura possa ser necessária. O projeto de Banco de Dados consiste em quatro etapas:

1. Análise de requisitos;
2. Modelagem conceitual;
3. Modelo lógico;

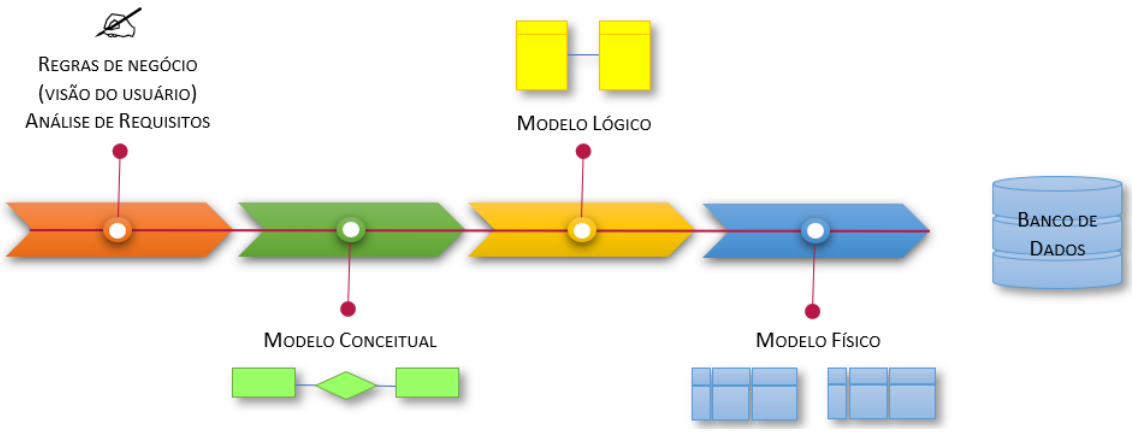
4. Modelo físico.

Aqui compartilhamos uma pequena dica com você: todo e qualquer projeto é guiado do início ao fim pelas regras de negócio, que representam as orientações e restrições que servem como reguladoras das operações de uma empresa. Essas regras são fornecidas pelo solicitante ou contratante.

Podemos citar alguns exemplos de regras de negócio:

- Uma pessoa para se matricular em um curso superior, obrigatoriamente, deverá apresentar o certificado de conclusão do Ensino Médio ou equivalente;
- Um candidato a ser motorista de aplicativo deverá possuir a carteira de habilitação válida.

Figura 3 – Fases de um projeto de Banco de Dados



Fonte: Albano; Albano, 2022.

3.2 Modelo conceitual

É a descrição do Banco de Dados de maneira independente do tipo de SGBD que será escolhido, ou seja, define quais os dados que precisam estar presentes no Banco de Dados, sem se importar com a implementação do modelo físico.

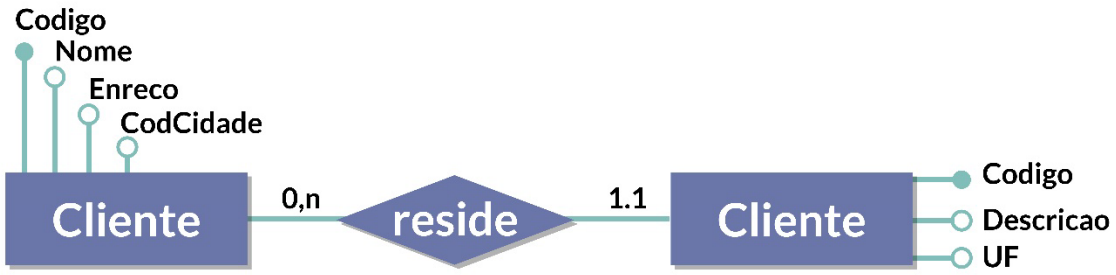
O modelo conceitual representa as regras de negócio sem preocupar-se com limitações tecnológicas ou de implementação, por isso, é a etapa mais adequada para o envolvimento do usuário que não precisa ter conhecimentos técnicos. Nesse modelo, temos:

- Visão geral do negócio;
- Facilitação do entendimento entre usuários e desenvolvedores;

- Somente as entidades e campos principais;
- Independente da implementação e do SGBD;
- Descrição mais abstrata do Banco de Dados;
- Ponto de partida para o projeto de um Banco de Dados.

Uma das técnicas mais utilizadas entre os profissionais da área é a abordagem Entidade-Relacionamento (ER), em que o modelo é representado por meio do Modelo Entidade-Relacionamento (MER).

Figura 4 – Exemplo de Modelo Entidade-Relacionamento (MER)



Fonte: Albano; Albano, 2022.

O modelo acima nos apresenta informações sobre as entidades Cliente e Cidade, em que para cada cliente será armazenado seu código de identificação, nome, endereço e código da cidade onde reside. Para cada cidade armazenaremos o código, o nome da cidade e a unidade federativa correspondente.

Vale salientar que, no exemplo acima, a relação ou associação existente entre as duas entidades se baseia na premissa que UM cliente reside apenas em UMA cidade, mas UMA cidade poderá ter NENHUM ou VÁRIOS clientes residentes. Essa relação é denominada *cardinalidade* e será abordada com mais detalhes em um próximo tópico desse material de estudo.

3.3 Modelo lógico

Nesta fase, descrevemos o Banco de Dados no nível do SGBD, ou seja, todo modelo considerará as limitações e as características particulares do tipo de tecnologia presente no Banco de Dados escolhido. Suas características são:

- Deriva do modelo conceitual;
- Define as chaves primárias das entidades;
- Normalização até a 3ª forma normal;

- Adequação ao padrão de nomenclatura;
- Relações e atributos documentados;
- Dependente do SGBD.

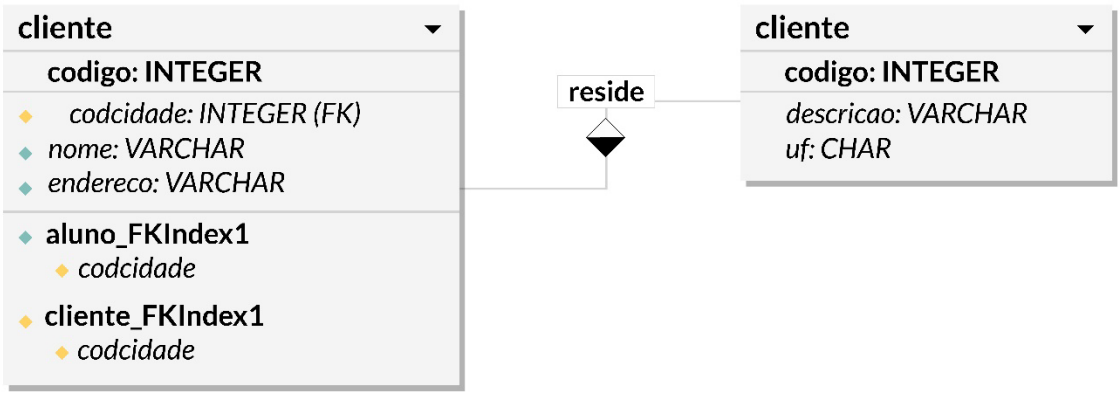
O modelo lógico do Banco de Dados relacional deve definir quais as relações (tabelas) e quais os atributos que compõem as relações, bem como os tipos de dados (número inteiro, data, cadeia de caracteres, entre outros) que serão armazenados nesses atributos.

Baseado no exemplo anterior, podemos definir o modelo lógico conforme:

- Cliente: codigo (integer), nome (varchar), endereco (varchar) e codcidade (integer);
- Cidade: codigo (integer), descricao (varchar) e UF (char).

É importante ressaltar que os detalhes internos de armazenamento não são descritos no modelo lógico, uma vez que essas informações fazem parte do modelo físico, que nada mais é que a tradução do modelo lógico para a linguagem do software escolhido para implementar o sistema.

Figura 5 – Exemplo de modelo lógico



Fonte: Albano; Albano, 2022.

3.4 Modelo Físico

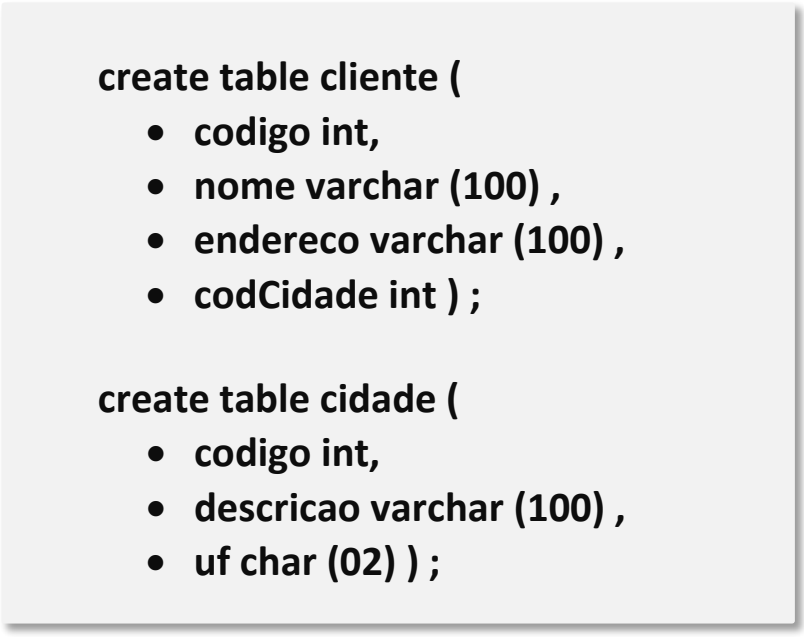
Considera os limites impostos pelo Sistema Gerenciador de Banco de Dados (SGBD) e pelos requisitos não funcionais dos programas que acessam os dados.

Características:

- Elaborado a partir do modelo lógico;

- Pode variar segundo o SGBD;
- Pode ter tabelas físicas (*log*);
- Descrição detalhada de como a base de dados está armazenada internamente;
- Linguagens e notações para o modelo físico variam de produto para produto (SGBD).

Figura 6 – Exemplo de modelo físico



Fonte: Albano; Albano, 2022.

TEMA 4 – MODELO ENTIDADE RELACIONAMENTO (MER)

O Modelo Entidade-Relacionamento (MER) foi desenvolvido pelo professor Peter Chen (1990) a fim de representar as estruturas de dados de uma forma mais natural e mais próxima do mundo real dos negócios.

O Modelo Entidade-Relacionamento (MER) é um modelo de dados conceitual de alto nível. Foi projetado para estar o mais próximo possível da visão que o usuário possui referente aos dados, não se preocupando em representar como esses dados estarão realmente armazenados.

O MER propõe que a realidade seja visualizada sob os seguintes conceitos fundamentais:

- Entidade: representação abstrata de um objeto do mundo real que desejamos armazenar informações e, que na maioria dos casos, irão

formar as tabelas do Banco de Dados. Por exemplo, informações sobre clientes, produtos, entregadores, restaurantes, entre outros;

- Campo: características particulares de cada entidade. Por exemplo, o cliente possui nome, data de nascimento, cidade onde reside, entre outras informações. O conjunto de informações contidas nos campos formam o que chamamos de registro;
- Relacionamento: representa a associação entre duas ou mais entidades. Por exemplo, o cliente reside em determinada cidade.

4.1 Entidade

Como já mencionamos, a entidade no modelo conceitual representa um objeto do qual desejamos armazenar as informações que serão utilizadas em uma aplicação.

Por exemplo, o sistema de delivery necessita armazenar informações sobre os clientes. Como existem diversas informações sobre cada cliente, nesse caso, se faz necessário criar uma entidade com a função de representar esse objeto.

4.1.1 Tipos de entidades

De acordo com a estrutura da chave primária e baseado no grau de dependência que uma entidade possui em relação às demais entidades do modelo, uma entidade poderá ser enquadrada como:

- Entidade fundamental;
- Entidade associativa;
- Entidade fraca.

Vale salientar que alguns dos conceitos citados acima serão esclarecidos após o aprofundamento dos conceitos do MER.

4.1.1.1 Entidade fundamental

É a entidade que possui chave primária simples, ou seja, a sua chave primária não é composta pela chave primária de nenhuma outra entidade. Dessa forma, a entidade fundamental não necessita da existência de qualquer outra entidade para existir, pois é independente.

Por exemplo, entidades Cliente, Produto, Entregador, Restaurante, entre outras.

4.1.1.2 Entidade associativa

É a entidade definida a partir da simplificação de um relacionamento de muitos para muitos entre duas ou mais entidades. Por exemplo, em um relacionamento entre a entidade Pedido e a entidade Produto, ocorre a seguinte situação: um pedido pode conter um ou vários produtos e um produto pode ser vendido em nenhum ou vários pedidos.

Com esse tipo de relacionamento surge a necessidade da criação de uma nova entidade (explicaremos melhor esse tipo de relacionamento no tópico sobre cardinalidade). Essa nova entidade poderá ser chamada de ItemPedido e nela serão armazenados alguns campos específicos, tais como quantidade vendida, valor do produto, desconto do produto, entre outros.

4.1.1.3 Entidade fraca

Essa entidade caracteriza-se pela dependência existencial, isto é, a entidade depende de uma outra entidade para poder existir no modelo.

Em casos em que exista a entidade fraca, tanto o relacionamento quanto a entidade deverão serem representados com borda dupla.

Por exemplo, em uma relação entre as entidades Funcionário e Dependente, sabemos que um funcionário poderá possuir nenhum ou vários dependentes. Dessa forma, a entidade Dependente apenas existe por causa da entidade Funcionário. Se eliminarmos a entidade Funcionário, consequentemente, a entidade Dependente desaparecerá devido a sua dependência.

Figura 7 – Exemplo de entidade fraca

Fonte: Albano; Albano, 2022.

4.1.2 Instância de uma entidade

São as informações armazenadas nos campos de uma entidade, as quais podem ser concretas (pessoa, conta, livro, entre outras) ou abstratas (pedido, transação, entre outras).

Instâncias do mesmo assunto são agrupadas sob uma mesma entidade. Por exemplo, o registro contendo as informações Wilquison Souza, CPF. 789.123.456.11, residente na Avenida 7 de Setembro, 1, caracteriza-se por ser uma instância da entidade Pessoa.

Figura 8 – Instância de uma entidade

PESSOA		
nome	cpf	endereco
Zanana Silva	123.456.789.00	Rua das Flores, 999
Wilquison Souza	789.123.456.11	Avenida 7 de setembro, 1

Fonte: Albano; Albano, 2022.

4.1.3 Regras para nomear uma entidade

- Deve ser única no modelo, isto é, não pode haver outra entidade com o mesmo nome;
- O nome não pode conter espaços em branco;
- O nome não pode conter caracteres especiais (ç, &, *, ˜, entre outros);
- Preferencialmente, opte por nomes curtos e significativos.

Exemplos de nomes de entidades: Cliente, Produto, Entregador, Restaurante, entre outros.

4.1.4 Forma de representação de uma entidade

As entidades são representadas no Modelo Entidade-Relacionamento (MER) por um retângulo. No centro do retângulo deve ser inserido o nome da entidade.

Figura 9 – Exemplo de representação de uma entidade

Fonte: Albano; Albano, 2022.

4.1.5 Estudo de caso

Para aprimorar os conceitos, vamos implementar passo a passo um estudo de caso sobre uma rede de restaurantes que decidiu ter um aplicativo de delivery.

O restaurante “Comer é Bom Demais” forneceu, inicialmente, as seguintes regras de negócio:

1. O restaurante possui diversas filiais espalhadas pelo território nacional, podendo ter mais de uma filial na mesma cidade. Essas informações devem estar devidamente cadastradas.
2. O restaurante possui vários funcionários.
3. Os pedidos são entregues por entregadores autônomos.
4. O aplicativo deverá possuir produtos.
5. Os pedidos referentes a cada restaurante deverão ser devidamente armazenados.
6. Para fazer um pedido no sistema de delivery, o cliente deverá estar previamente cadastrado.

Vale ressaltar que, ao longo do desenvolvimento do modelo, serão incluídas novas regras de negócio com o objetivo de aprofundar e aplicar os conceitos aqui discutidos.

Para começar, vamos definir as entidades que representaram cada objeto do mundo real da empresa de delivery, baseando-se sempre nas regras de negócio já conhecidas.

Avalie as regras de negócio perguntando-se sobre quais objetos ou itens têm características ou informações próprias a serem armazenadas. Pensando dessa forma, já podemos identificar as seguintes entidades:

- Como a empresa atua em âmbito nacional, seria importante armazenarmos informações sobre os **estados** e as **cidades**;

- O quadro de **funcionários** dessa empresa também precisa ser contemplado, pois a empresa necessita dessas informações;
- **Clientes**, **restaurantes**, **entregadores** e **produtos** seguem a mesma linha de raciocínio;
- Quanto a movimentação dos **pedidos**, a empresa precisa ter um controle do que foi vendido, para quem foi vendido e qual restaurante vendeu.

Perceba que cada objeto que armazena informações foi definido como uma *entidade*. É possível que você tenha encontrado outras entidades que não foram listadas nesse momento. Não se preocupe, durante a evolução do modelo é comum a inclusão e a exclusão de entidades até chegarmos ao modelo ideal. Isso faz parte do processo de construção do projeto de Banco de Dados.

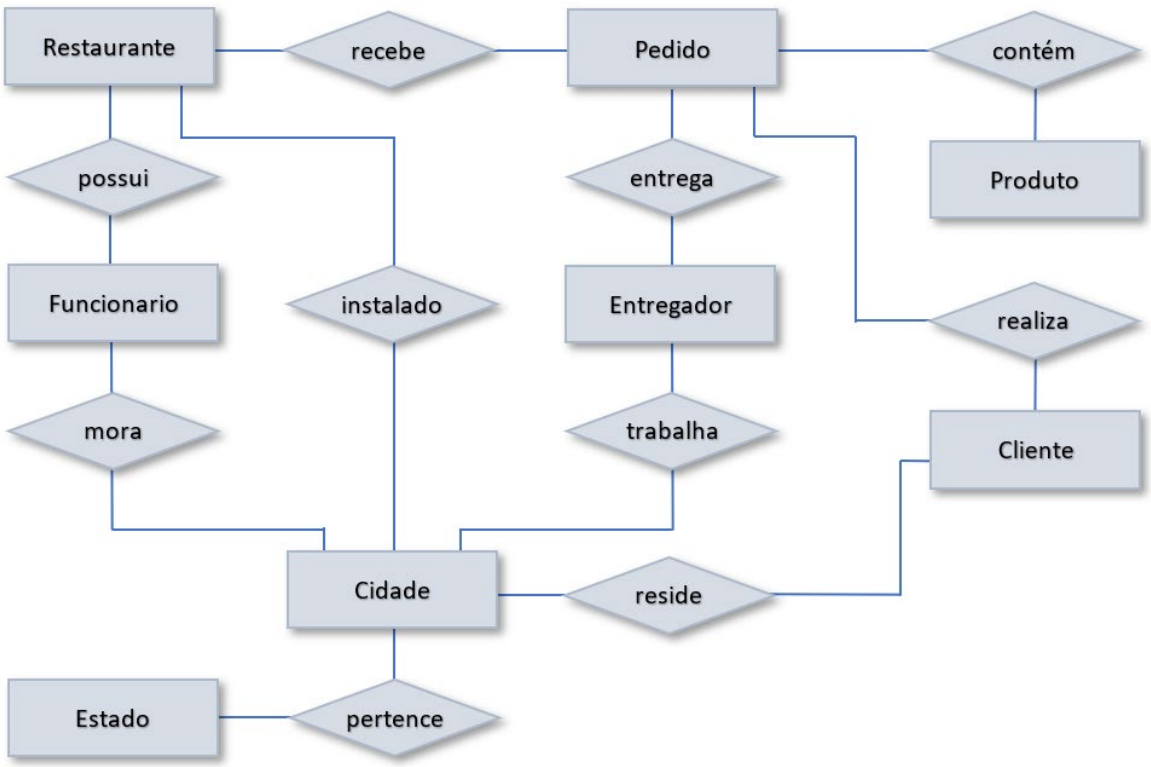
Baseado nas entidades que já descobrimos, vamos analisar qual a relação que cada uma pode possuir com as demais:

- Restaurante:
  - Possui várias filiais que estão instaladas em diferentes cidades;
  - Cada filial receberá vários pedidos por meio do aplicativo;
  - Possui muitos funcionários em cada filial;
- Cliente:
  - Realiza um pedido por meio do aplicativo;
  - Reside em uma cidade;
- Funcionário:
  - Faz parte do quadro de funcionários de uma das filiais;
  - Reside em uma cidade;
- Entregador:
  - Realiza as entregas dos pedidos;
  - Está locado em uma cidade;
- Pedido:
  - Precisa-se manter as informações sobre quem fez o pedido, para qual filial e qual(is) foi(ram) o(s) produto(s) escolhido(s);
- Produto:
  - Sempre será referenciado dentro de um pedido;
- Cidade:
  - Pertence a um estado;
- Estado:

- Possui diversas cidades em seu território.

A partir dessa análise prévia, vamos construir nosso primeiro modelo conceitual (MER).

Figura 10 – Estudo de caso: Primeiro modelo conceitual



Fonte: Albano; Albano, 2022.

O próximo passo é aprendermos sobre os campos que pertencem a cada uma das entidades do nosso modelo conceitual.

4.2 Campos

São as propriedades que caracterizam uma entidade, isto é, características particulares do objeto que está sendo analisado. Os campos são baseados nas informações definidas nas regras de negócio.

No caso do nosso exemplo sobre o sistema de delivery, a entidade Cliente possuirá como campos o nome, o endereço, a data de nascimento, o sexo, o CPF e as demais características necessárias.

4.2.1 Tipos de campos

De acordo com a sua finalidade ou conteúdo, um campo pode ser classificado da seguinte forma:

- Obrigatório;
- Opcional;
- Simples;
- Composto;
- Monovalorado;
- Multivalorado;
- Derivado.

4.2.1.1 Campo obrigatório

É aquele campo que deve possuir, para uma instância de uma entidade ou relacionamento, obrigatoriamente um valor, ou seja, o campo deve ter um conteúdo, não podendo ficar sem informação. Por exemplo, nome do cliente, descrição do produto, nome da cidade, entre outros.

4.2.1.2 Campo opcional ou nulo

É aquele que para uma instância da entidade ou relacionamento pode possuir um valor, ou seja, seu preenchimento não é obrigatório. Por exemplo, telefone, complemento do endereço, entre outros.

4.2.1.3 Campo simples ou atômico

São os campos que não podem ter o seu conteúdo dividido, ou seja, não há como separar os dados. Campos desse tipo não são divisíveis ou decompostos. Por exemplo, estado civil, bairro, gênero, raça, entre outros.

4.2.1.4 Campo composto

Os campos compostos podem ser divididos em partes menores que, dependendo do caso, podem inclusive gerar novos campos.

O próprio nome de uma pessoa pode ser classificado como um campo composto, pois pode ser estruturado em prenome, nome intermediário e sobrenome.

4.2.1.5 Campo monovalorado

Todo e qualquer campo que possua um único valor para uma entidade é considerado um campo monovalorado. Por exemplo, nome, CPF, gênero, data de nascimento, entre outros.

4.2.1.6 Campo multivalorado

Os campos que podem possuir mais de um valor para uma entidade são chamados de *campos multivalorados*. Por exemplo, uma pessoa pode possuir mais de um número de telefone.

4.2.1.7 Campo derivado

Um campo derivado é aquele que é derivado de outros campos ou entidades relacionados a ele. Por exemplo, o campo idade é derivado dos campos data de nascimento e data atual.

4.2.2 Domínio do campo

Cada campo possui um domínio, que é um conjunto de valores que pode representá-lo. O domínio é um conjunto de valores que possuem determinadas propriedades em comum. Ao conjunto de todos os valores possíveis para determinado campo dá-se o nome de *domínio*. Exemplo:

- Código do cliente: o domínio é definido como um conjunto de números inteiros positivos com quatro algarismos;
- Nome do entregador: conjunto de caracteres alfanuméricos com no máximo 100 caracteres;
- Salário: valor numérico positivo com duas casas decimais;
- Gênero: são os mnemônicos M ou F.

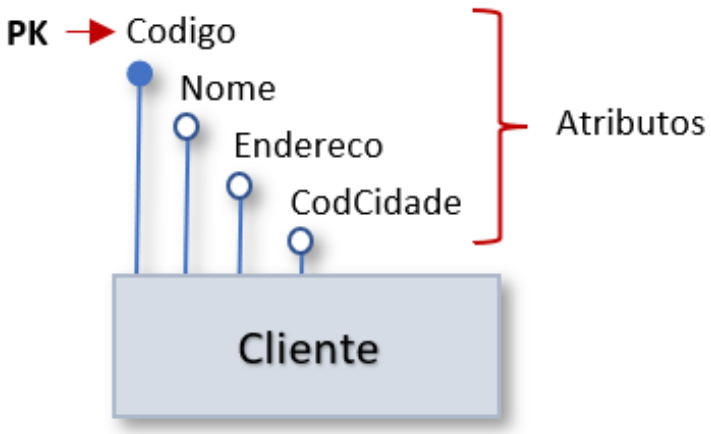
4.2.3 Regras para nomear um campo

- Devem ser únicos na entidade;
- Nomes sem espaços em branco;
- Nomes sem caracteres especiais;
- Utilizar nomes significativos que identifiquem o seu conteúdo;
- Preferencialmente nomes curtos.

4.2.4 Formas de representação de um campo

No Modelo Entidade-Relacionamento (MER) o campo é representado por meio de uma linha que o liga a uma determinada entidade que o corresponde. Essa linha também possui uma ponteira que representa a prioridade que o campo exerce dentro da entidade, podendo ser um campo principal da entidade, denominado *chave primária* ou *Primary Key* (PK, o único campo a ser representado pela ponteira preenchida), campos que assumem as funções de chave secundária (*Secondary Key*), estrangeira (*Foreign Key* – FK) ou, simplesmente, um campo comum.

Figura 11 – Exemplo de representação de campos



Fonte: Albano; Albano, 2022.

4.2.5 Tipos de chaves de um campo

Uma entidade deve ter a capacidade de identificar cada uma de suas instâncias separadamente em um Banco de Dados, além de manter a capacidade de relacionar-se com as demais entidades. Para isso, utilizamos o conceito de *chaves*.

4.2.5.1 Chave primária ou *Primary Key*

Uma chave primária é um ou um conjunto de campos que permite identificar unicamente uma instância da entidade. Tem por função diferenciar uma instância da entidade de outra instância, tornando essa instância única, ou seja, diferente das demais instâncias.

Características de uma chave primária:

- Identificada pela sigla PK (*Primary Key*);
- O valor contido nesse campo deve ser único para cada instância da entidade, ou seja, nunca se repetirá. Além disso, não poderá sofrer alteração, garantindo que cada instância possua a característica de unicidade;
- Não poderá receber valores duplicados ou nulos. Dessa forma, também não poderá ser composta por um campo opcional;
- Chave primária que apresente mais de um campo é denominada de *chave primária composta*;
- Preferencialmente, a chave primária deve ser um campo numérico, uma vez que isso ajudará na performance do Banco de Dados.

Chave primária não natural: existem situações em que não temos um campo “natural”, ou seja, próprio da entidade que possa ser a chave primária. Nesse caso, usaremos o conceito de chave substituta ou chave artificial (*Surrogate Key*). Geralmente, esse campo numérico é sequencial (autoincremento), sendo sua única função diferenciar uma instância de outra.

Para melhor compreensão, vamos exemplificar a definição de uma chave primária utilizando a entidade Cliente. Procure analisar cada campo da entidade a seguir, levando em conta as regras já discutidas anteriormente. Pergunte-se: qual o campo (coluna) ideal para ser eleito como chave primária?

Figura 12 – Exemplo de chave primária

CLIENTE			
codigo	nome	nascimento	endereco
20001001	João da Silva	01/01/1980	Sete de setembro, 1000
20002100	Maria de Souza	30/01/1985	XV novembro, 10
19992009	Paulo de Gil	01/01/1980	General Osório, 102
20039564	Maria de Souza	12/10/1990	Getúlio Vargas, 200
20093212	Ana de Jesus	26/09/1992	Sete de setembro, 1000

Fonte: Albano; Albano, 2022.

Vamos analisar a entidade (tabela):

- As colunas que contêm, respectivamente, o nome do cliente, a data de nascimento e o endereço não armazenam um valor único que represente o registro, isso porque podem existir homônimos (pessoas com o mesmo nome), pessoas que nascem na mesma data e, para concluir, em um mesmo endereço pode morar uma família;
- Sendo assim, a melhor escolha para ser a chave primária é o campo código, uma vez que cada cliente possui um número que difere de cliente para cliente, o que torna seu conteúdo único, inalterável e intransferível. Perceba que o campo código não é um campo natural da entidade Cliente, ele existe pela necessidade de diferenciar um cliente de outro cliente.

4.2.5.2 Chave estrangeira ou *Foreign Key*

A chave estrangeira é um campo ou um conjunto de campos presentes em uma entidade, mas que pertencem a outra entidade. Além disso, esse campo deverá ser uma chave primária na sua entidade de origem.

Vale salientar que a presença de chaves estrangeiras em uma entidade caracteriza a associação entre entidades, isto é, só existe chave estrangeira se houver um relacionamento direto entre as entidades envolvidas.

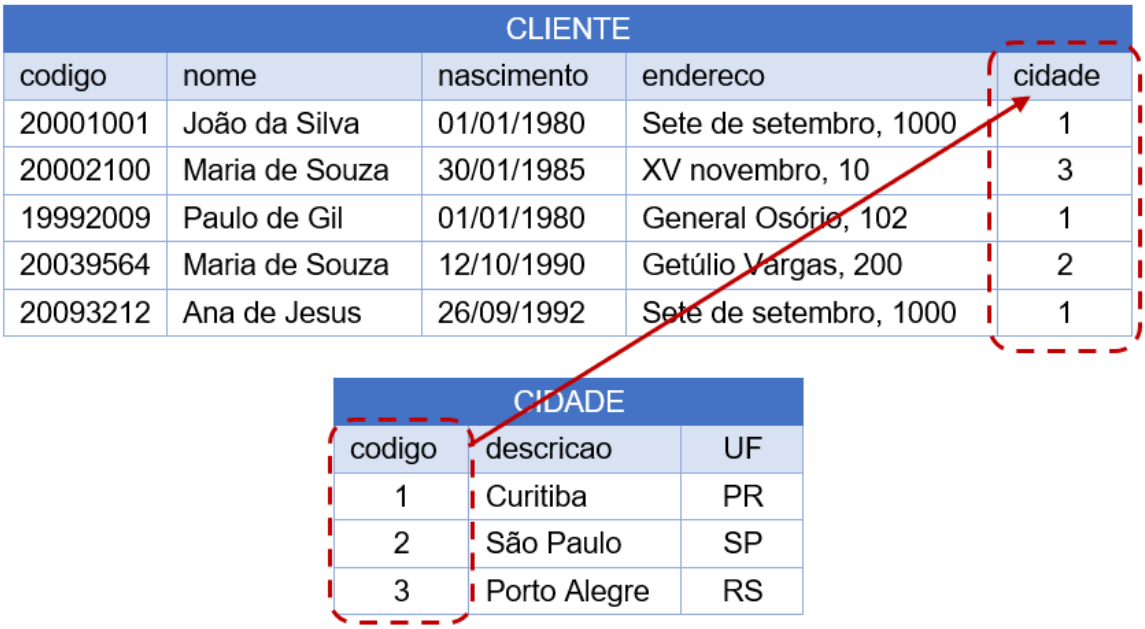
Características de uma chave estrangeira:

- Identificada pela sigla FK (*Foreign Key*);
- Utilizada para referenciar o relacionamento com outra entidade;
- Sempre será uma chave primária de outra entidade associada;

- Caso a chave primária seja composta na origem, a chave estrangeira também será composta;
- O valor da chave estrangeira poderá se repetir;
- Não poderá ser composta por campo opcional, ou seja, campo que aceite um valor nulo.

Exemplo:

Figura 13 – Exemplo de chave estrangeira



CLIENTE				
codigo	nome	nascimento	endereco	cidade
20001001	João da Silva	01/01/1980	Sete de setembro, 1000	1
20002100	Maria de Souza	30/01/1985	XV novembro, 10	3
19992009	Paulo de Gil	01/01/1980	General Osório, 102	1
20039564	Maria de Souza	12/10/1990	Getúlio Vargas, 200	2
20093212	Ana de Jesus	26/09/1992	Sete de setembro, 1000	1

CIDADE		
codigo	descricao	UF
1	Curitiba	PR
2	São Paulo	SP
3	Porto Alegre	RS

Fonte: Albano; Albano, 2022.

Analisando a entidade Cliente podemos verificar que ela possui um campo que armazenará a cidade onde reside cada cliente. Essa informação, porém, tem uma característica repetitiva, em que é possível que a mesma cidade seja usada em diversos registros de clientes. Dessa forma, para garantir agilidade e precisão nos dados informados foi criada uma nova entidade chamada Cidade.

No caso da entidade Cidade, ela conterá todos os registros de cada uma das cidades e, dessa forma, a entidade Cliente apenas irá fazer a referência ao código correspondente a cidade desejada.

Perceba que as duas entidades se comunicam apenas por meio do código da cidade, que é a chave estrangeira dessa relação.

4.2.6 Estudo de caso

Baseado no conhecimento adquirido sobre campos, vamos continuar o desenvolvimento do nosso projeto de Banco de Dados sobre um aplicativo de delivery.

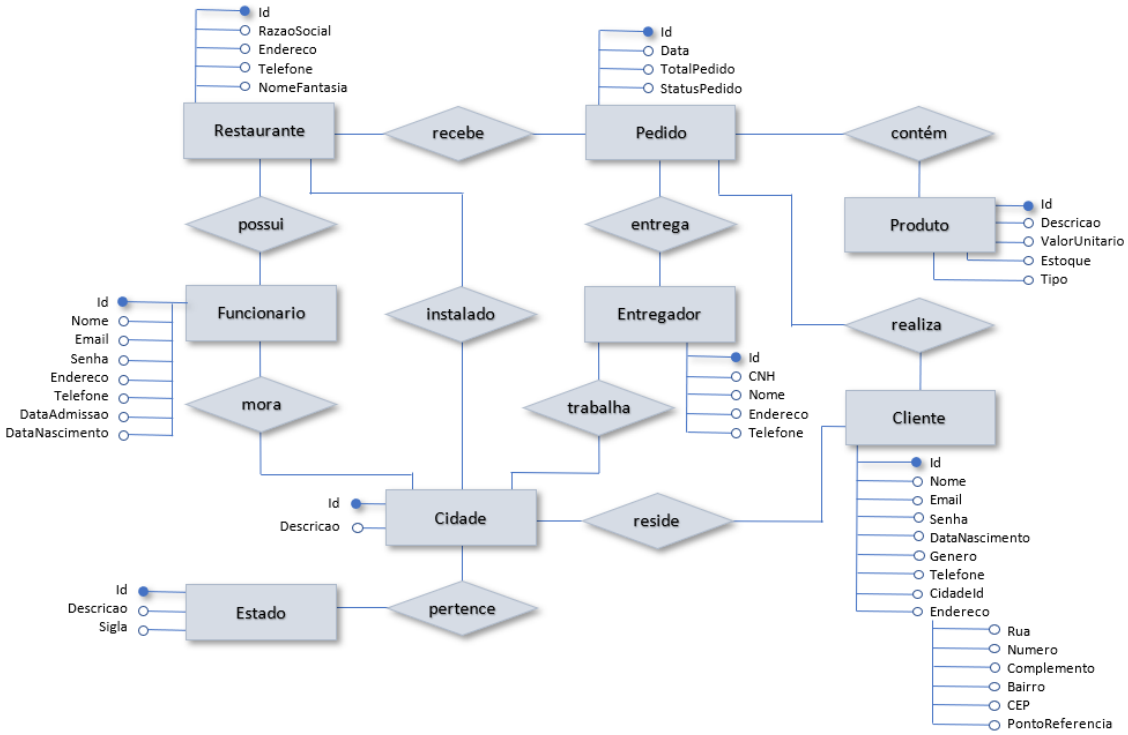
Regras adicionais:

- O acesso ao aplicativo, tanto por clientes quanto por funcionários, deverá ser por meio de um login e uma senha, sendo que o login corresponde ao e-mail do usuário;
- Como é um sistema de delivery, o endereço de entrega deve ser bem detalhado, contemplando as seguintes informações: nome da rua, número do imóvel, complemento (caso houver), bairro, ponto de referência e CEP;
- O restaurante sempre presenteia os aniversariantes com um desconto;
- Os pedidos possuem um status, isto é, a situação em que se encontra, podendo estar no estado “A confirmar”, “Em preparação”, “Saiu para a entrega”, “Entregue” e “Cancelado”;
- Os entregadores autônomos, obrigatoriamente, deverão ter registrada a carteira de habilitação no sistema;
- Os produtos são divididos em categorias, sendo: lanches, pratos prontos, bebidas e sobremesas.

Vale salientar que, além das regras adicionais, alguns campos se fazem necessários, por exemplo, informações do cliente, do produto, do funcionário, entre outros.

Importante ressaltar que o modelo conceitual apresentado não contempla os campos resultantes dos relacionamentos entre as entidades. Tais campos serão incluídos no momento da aplicação das regras de cardinalidade.

Figura 14 – Inclusão de campos no modelo conceitual



Fonte: Albano; Albano, 2022.

4.3 Relacionamento ou associação

Um relacionamento pode ser entendido como uma associação entre as entidades devido as regras de negócio. Normalmente, ocorre entre duas ou mais entidades, porém, há situações que pode ocorrer entre instâncias da mesma entidade, conhecido como *autorrelacionamento*.

Por exemplo, um cliente (entidade Cliente) reside em determinada cidade (entidade Cidade). Dessa forma, estabelece-se uma ligação, isto é, um relacionamento entre as entidades Cliente e Cidade.

4.3.1 Forma de representação

Para representar a ocorrência de relacionamentos utilizamos um losango que estabelece a ligação entre as entidades envolvidas, conectando-as por meio de uma linha.

Figura 15 – Representação de relacionamento

Fonte: Albano; Albano, 2022.

Por exemplo, a entidade Cliente relaciona-se com a entidade Cidade, onde um cliente reside em uma cidade e uma cidade pode ter nenhum ou vários clientes residentes.

Figura 16 – Exemplo de relacionamento

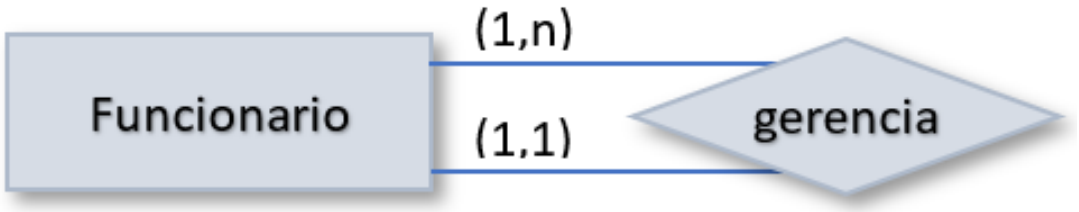
Fonte: Albano; Albano, 2022.

4.3.2 Relacionamento recursivo ou autorrelacionamento

Os relacionamentos recursivos, também chamados de *autorrelacionamentos*, são casos especiais em que uma entidade se relaciona consigo mesma, isto é, ocorre um relacionamento associado a uma única entidade.

Podemos exemplificar esse tipo de relacionamento utilizando o gerenciamento de funcionários de uma empresa, em que o gerente é um funcionário que possui um relacionamento com outros funcionários que lhe são subordinados. Porém, o gerente também é um funcionário da empresa e subordinado a outro funcionário que ocupa um cargo superior ao de gerente. Dessa forma, seu papel como funcionário oscila entre gerente e subordinado. Esse relacionamento pode ser representado da seguinte forma:

Figura 17 – Representação de relacionamento recursivo



Fonte: Albano; Albano, 2022.

O autorrelacionamento pode possuir uma cardinalidade do tipo 1:1 (um para um), 1:n (um para muitos) ou n:n (muitos para muitos), dependendo da política de negócio que estiver envolvida. O tema cardinalidade será discutido posteriormente nesse material de estudo.

4.3.3 Especialização e generalização

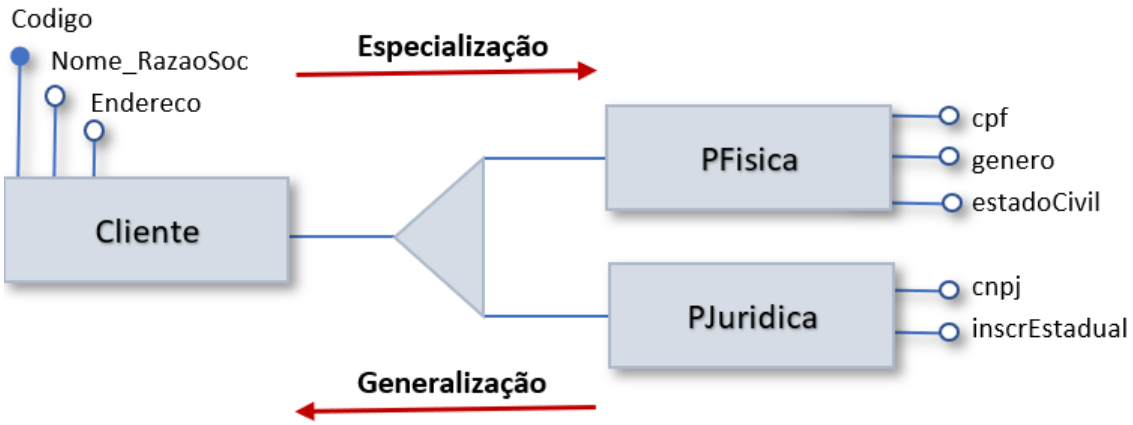
A especialização ocorre quando existe a presença de propriedades particulares em uma entidade, provocando uma subdivisão devido a necessidade de campos específicos para cada situação.

No caso da generalização, essa é exatamente o inverso da especialização, isto é, agregando as similaridades presentes das especializações. É quando as entidades podem ser reunidas em apenas uma.

O clássico exemplo do cliente pessoa física ou pessoa jurídica ilustra claramente como ocorre uma situação de especialização. Analisando o conjunto de campos é possível verificar que uma pessoa física possui campos diferentes de uma pessoa jurídica, mas ambos são clientes.

Veja o modelo a seguir:

Figura 18 – Exemplo de especialização e generalização



Fonte: Albano; Albano, 2022.

TEMA 5 – CARDINALIDADE

A cardinalidade é um elemento fundamental no relacionamento entre entidades de um modelo. É por meio dela que determinamos se a relação entre as entidades está correta. A cardinalidade é baseada em suas instâncias. Além disso, a cardinalidade define em qual entidade será adicionada a chave estrangeira (*Foreign Key* – FK).

Lembre-se, havendo um relacionamento, teremos um ou mais campos em comum entre as entidades envolvidas, que são responsáveis pela ligação entre elas.

De forma básica, a cardinalidade é representada de três formas:

1. Um para um (1:1);
2. Um para muitos / muitos para um (1:n) / (n:1);
3. Muitos para muitos (n:n) / (n:m).

Além dos tipos citados acima, também abordaremos a cardinalidade máxima e mínima, extremamente utilizadas.

5.1 Um para um (1:1)

Ocorre quando UMA instância da entidade está associada a apenas UMA instância de outra entidade e, de forma inversa, o mesmo acontece.

Por exemplo, baseado na regra de negócio que uma pessoa só pode estar casada legalmente com outra pessoa (cônjuge), como estabeleceríamos a cardinalidade existente entre as entidades?

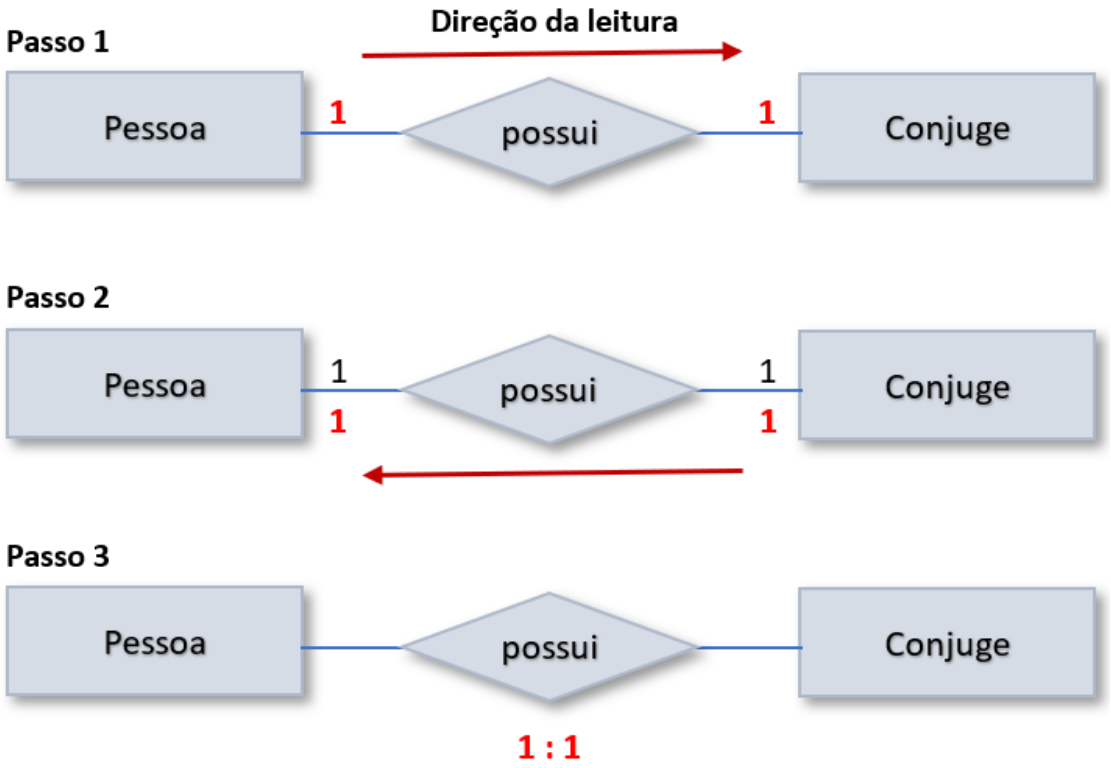
Figura 19 – Exemplo de relacionamento

Fonte: Albano; Albano, 2022.

Passo a passo:

1. UMA pessoa (1) possui apenas UM cônjuge (1).
2. Avalie a cardinalidade obtida realizando a leitura de forma inversa: UM cônjuge (1) possui apenas UMA pessoa (1).
3. Perceba que a cardinalidade permaneceu igual, não importando a direção da leitura. Dessa forma, a cardinalidade resultante é 1:1.

Figura 20 – Exemplo de cardinalidade 1:1

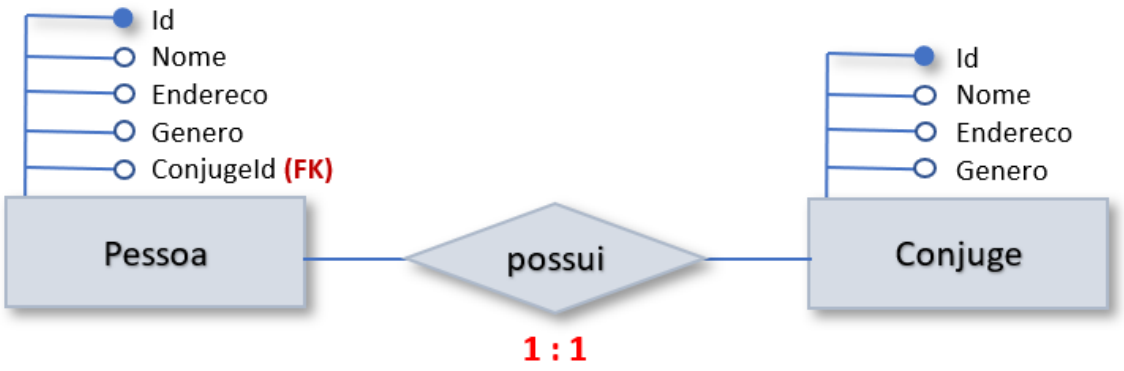


Fonte: Albano; Albano, 2022.

5.1.1 Definição da Chave Estrangeira – *Foreign Key* (FK)

Em um relacionamento 1:1 a chave estrangeira poderá ficar em qualquer uma das entidades. Como é um relacionamento exclusivo (1:1), não faz diferença se a chave primária da entidade Pessoa for adicionada na entidade Cônjuge ou de forma inversa.

Figura 21 – Chave estrangeira em cardinalidade 1:1



Fonte: Albano; Albano, 2022.

Vale ressaltar que quando ocorrer um relacionamento 1:1, sua existência é opcional, ou seja, analisando as entidades envolvidas no relacionamento você pode decidir eliminar uma delas. No exemplo, a entidade Cônjuge poderia ser transformada em um campo da entidade Pessoa. Eliminar ou não a entidade é uma decisão pessoal de quem está modelando, não havendo uma regra definida.

5.2 Um para muitos / muitos para um (1:n / n:1)

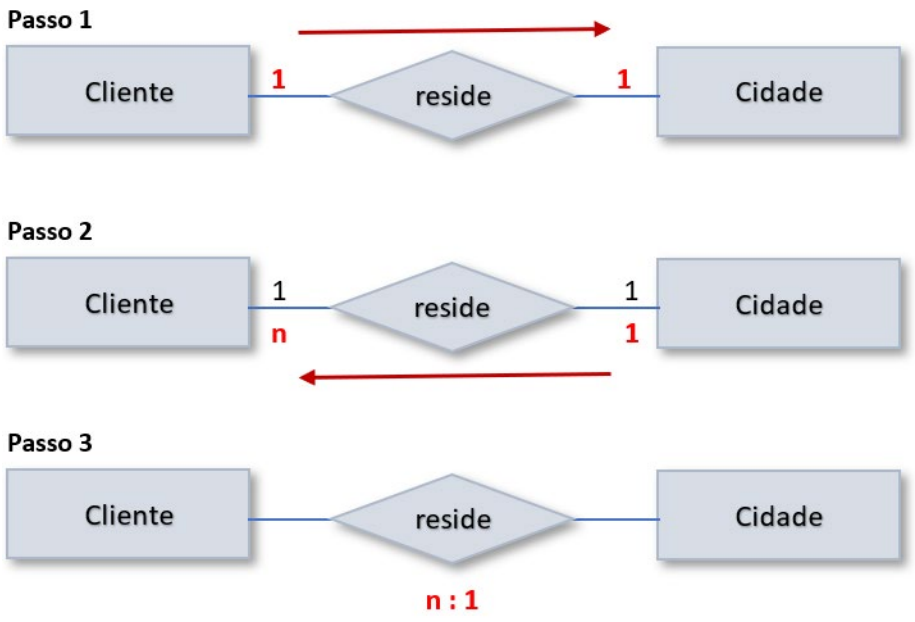
Uma entidade está associada a várias instâncias de outra entidade, mas de forma inversa, uma entidade apenas pode estar associada a no máximo UMA instância da entidade. Também é importante ressaltar que a cardinalidade n sempre terá precedência sobre a cardinalidade 1.

Por exemplo, um cliente está associado unicamente a uma cidade, mas o mesmo não ocorre de forma inversa, pois uma cidade possui vários clientes associados a mesma.

Passo a passo:

1. UM cliente (1) reside em apenas UMA cidade (1).
2. Leitura inversa: UMA cidade (1) possui MUITOS residentes (n).
3. Perceba que a leitura da cardinalidade apresentou diferenças. Por causa disso, devemos aplicar a regra da precedência do n em relação a cardinalidade 1, resultando em uma cardinalidade de 1:n.

Figura 22 – Exemplo de cardinalidade 1:n



Fonte: Albano; Albano, 2022.

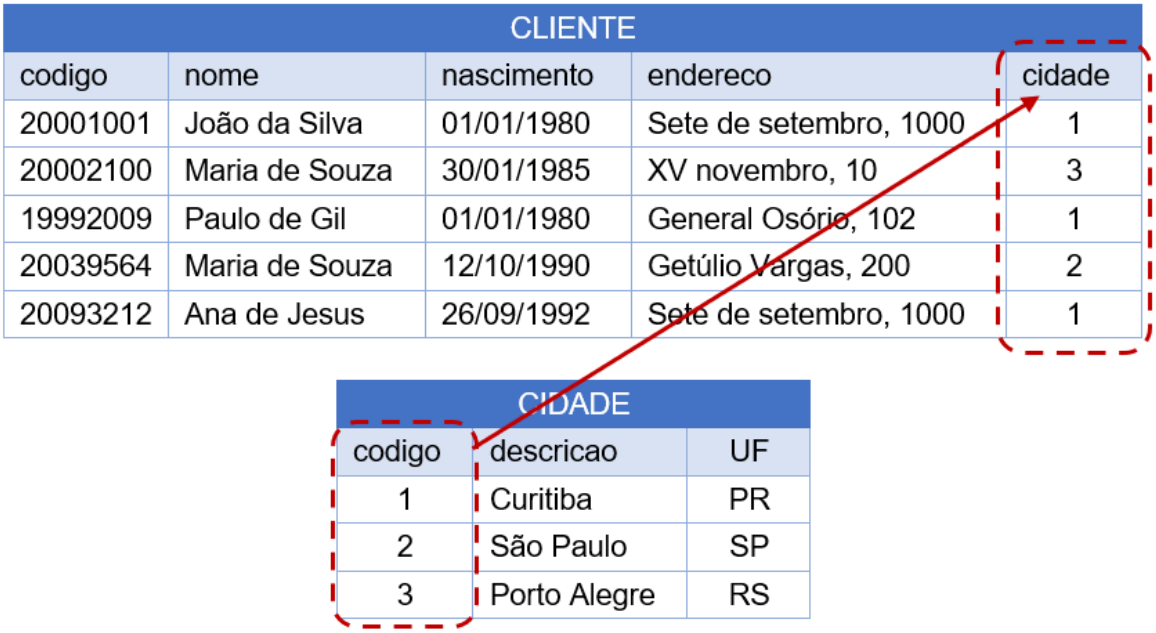
5.2.1 Definição da chave estrangeira – *Foreign Key* (FK)

Em um relacionamento 1:n ou n:1, a chave estrangeira deverá ficar sempre na entidade do lado n do relacionamento.

No exemplo, a chave primária da entidade Cidade será adicionada na entidade Cliente. Nessa entidade, esse campo será a chave estrangeira, ou seja, será a ligação entre as entidades Cliente e Cidade.

Você pode estar se perguntando: por que não colocamos a chave estrangeira (FK) no lado do relacionamento 1? Essa não seria uma boa opção, pois teríamos que armazenar na entidade Cidade inúmeros clientes, o que não seria muito prático ou viável. Sendo assim, a chave estrangeira fica na entidade que terá somente um valor a ser associado, ou seja, um cliente reside somente em uma cidade.

Figura 23 – Exemplo de chave estrangeira



Fonte: Albano; Albano, 2022.

5.3 Muitos para muitos (n:n / n:m)

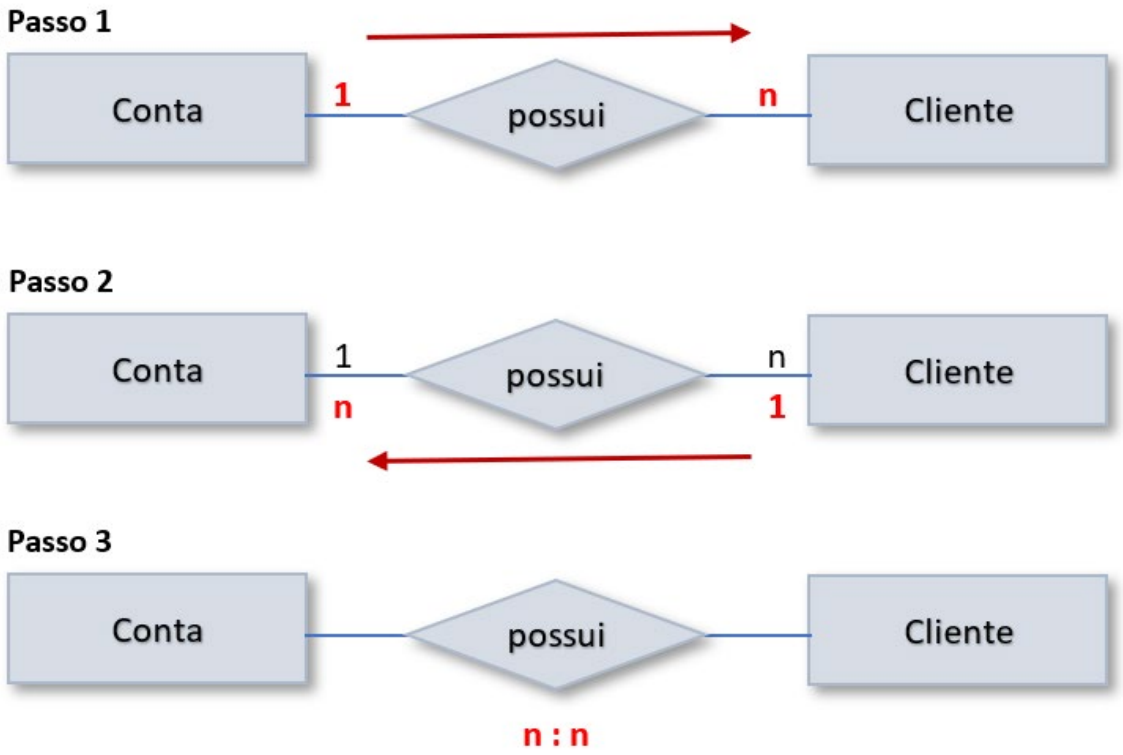
Neste caso, uma entidade está associada a várias instâncias de outra entidade e o inverso ocorre da mesma maneira.

Por exemplo, considere a regra de negócio que define que um cliente poderá possuir várias contas e que uma conta poderá ser conjunta, isto é, pertencer a mais de um cliente.

Passo a passo:

1. UMA conta (1) possui MUITOS clientes (n).
2. Leitura inversa: UM cliente (1) pode ter MUITAS contas (n).
3. Aplique a regra da precedência e a cardinalidade resultante é n:n.

Figura 24 – Exemplo de cardinalidade n:n



Fonte: Albano; Albano, 2022.

5.3.1 Definição da chave estrangeira – *Foreign Key* (FK)

Este caso é o mais complexo para analisarmos. Já estudamos que a regra define que a chave estrangeira (FK) é indicada no lado n, porém, nesse caso, os dois lados resultaram em uma cardinalidade muitos para muitos (n:n). Dessa forma, precisamos ponderar sobre algumas questões:

- Podemos adicionar a chave primária da entidade Conta na entidade Cliente?
- Ou adicionar a chave primária da entidade Cliente na entidade Conta?

A resposta para as duas questões é não, pois em ambos os casos teríamos que armazenar mais de um valor em um mesmo campo. Então, qual é a solução?

Relacionamentos cuja cardinalidade resulte em n:n sempre gerarão uma nova entidade. Essa entidade, como já vimos, é conhecida como *entidade associativa*. Nela adicionaremos a chave primária das entidades envolvidas no relacionamento, nesse caso, as entidades Conta e Cliente. Inclusive, se houver necessidade, podemos adicionar mais campos.

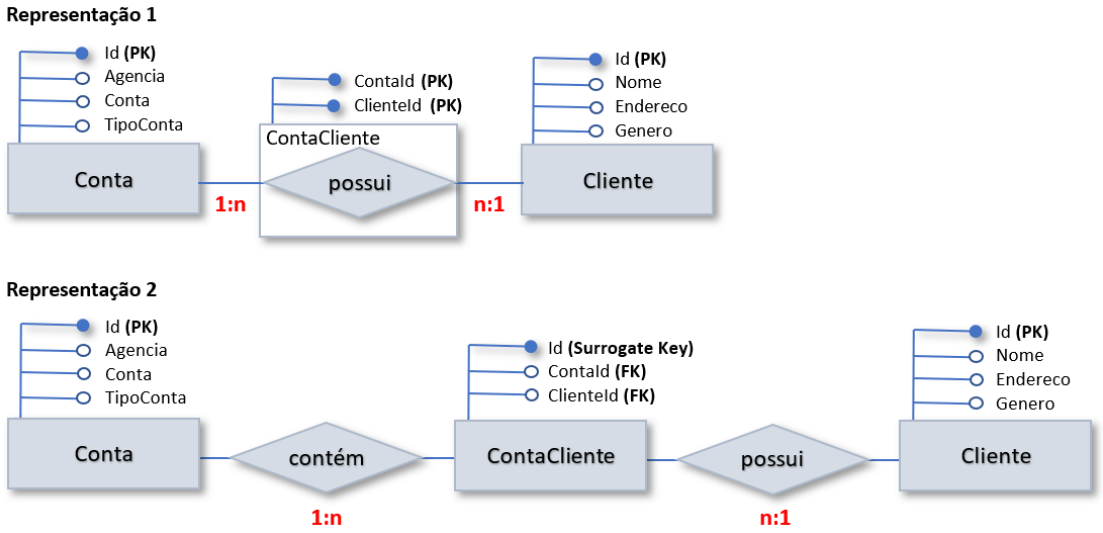
A partir do momento em que ocorre a inclusão dessa nova entidade, o relacionamento torna-se automaticamente 1:n.

Outro aspecto importante nessa situação que precisamos avaliar é: qual será a chave primária dessa nova entidade?

Existem duas alternativas para a definição dessa chave:

1. A chave primária será composta pelas chaves estrangeiras das entidades Conta e Cliente. Isso mesmo que você está lendo, um campo pode ser chave estrangeira e fazer parte da chave primária ao mesmo tempo.
2. Criar uma chave artificial, denominada *Surrogate Key* (chave substituta), que será um campo com a função de diferenciar uma instância da outra.

Figura 25 – Formas de representação de uma entidade associativa



Fonte: Albano; Albano, 2022.

Com a prática na criação de modelos, você irá naturalmente antever um relacionamento n:n e já criará as entidades sem a necessidade de representar a entidade associativa (representação 2).

Observe que na representação 2 da entidade ContaCliente foi utilizado como chave primária uma *Surrogate Key* (SK), em vez da chave composta apresentada na representação 1.

Saiba mais

Após finalizar a aplicação da cardinalidade no Modelo Entidade-Relacionamento (MER), não poderá existir nenhum relacionamento n:n. Todos os relacionamentos resultantes deverão apresentar cardinalidades 1:n e 1:1.

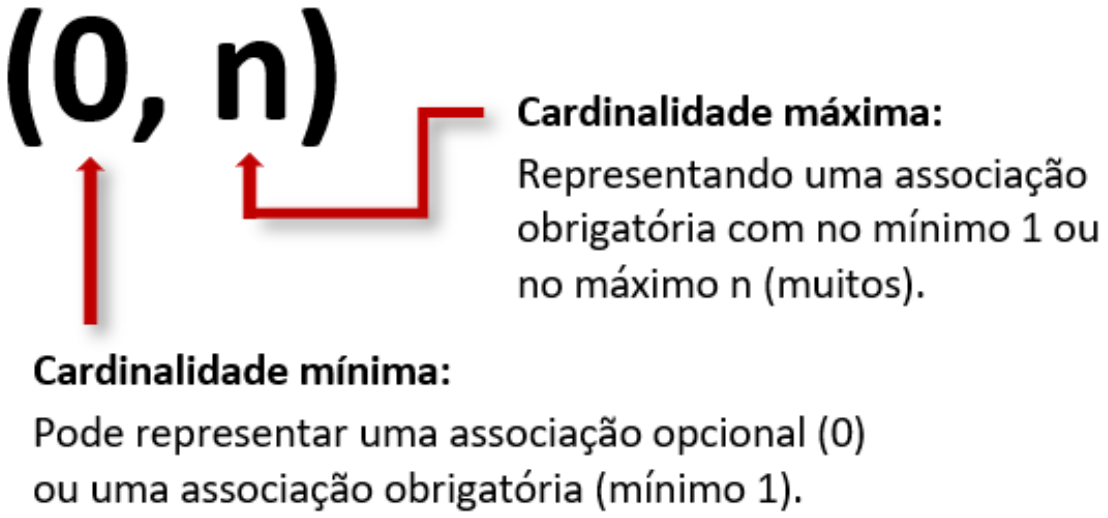
5.4 Cardinalidades mínima e máxima

Com a evolução do modelo conceitual (MER) e com o objetivo de melhorar a implementação das regras de negócio, foi introduzido o conceito de cardinalidades mínima e máxima. Tal conceito representa a frequência de relações entre as entidades, isto é, restringe a quantidade de ocorrências que podem existir entre as entidades que participam do relacionamento. Essas ocorrências também são denominadas *instâncias*.

A cardinalidade é representada por uma cardinalidade mínima e uma cardinalidade máxima, possuindo uma notação (m, M), onde:

- Cardinalidade mínima (m) – Estabelece o número mínimo de instâncias que podem existir entre uma entidade e outras entidades participantes do relacionamento;
- Cardinalidade máxima (M) – Define o número máximo de instâncias que podem existir entre uma entidade e outras entidades envolvidas no relacionamento.

Figura 26 – Cardinalidades mínima e máxima

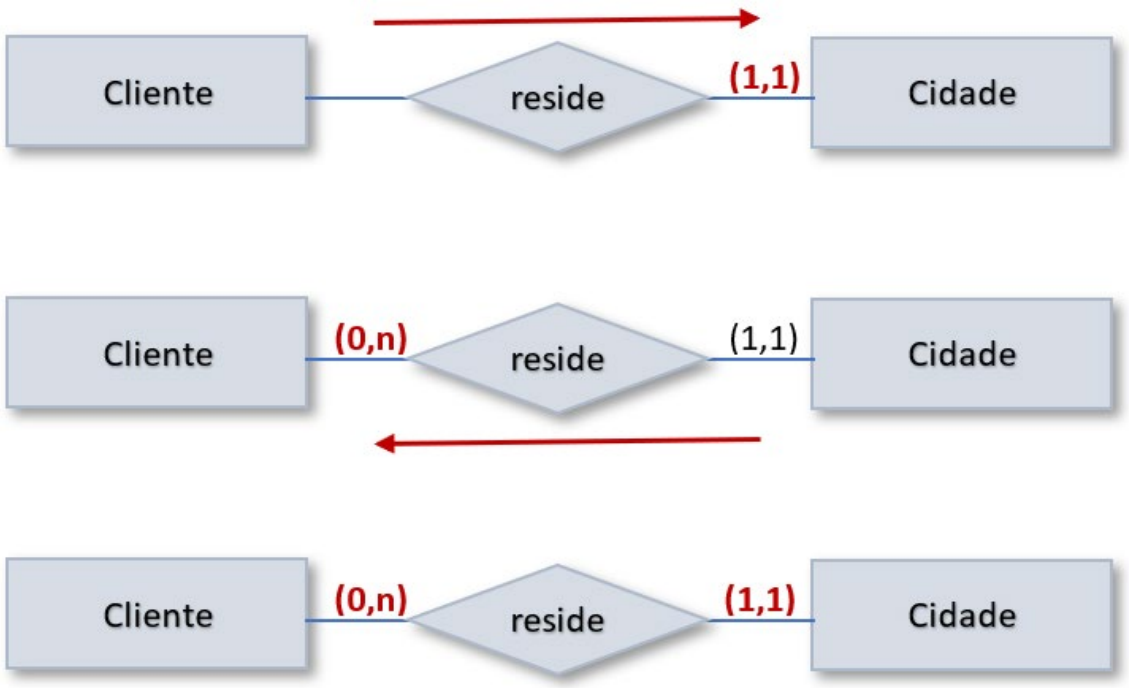


Fonte: Albano; Albano, 2022.

É importante ressaltar que a notação da cardinalidade é sempre apresentada no lado oposto. Por exemplo, na relação existente entre as entidades Cliente e Cidade:

1. UM cliente reside em apenas UMA cidade. Dessa forma, tanto a cardinalidade mínima quanto a máxima serão 1, sendo a cardinalidade resultante (1,1). A notação será apresentada ao lado da entidade Cidade.
2. Leitura inversa: UMA cidade poderá ter NENHUM (zero) ou MUITOS residentes associados a ela, resultando em uma cardinalidade (0,n). A notação será apresentada ao lado da entidade Cliente.

Figura 27 – Exemplo de cardinalidades mínima e máxima



Fonte: Albano; Albano, 2022.

É importante ressaltarmos que, independentemente da ferramenta utilizada, a cardinalidade possui mais de um padrão de notação. A seguir apresentaremos as mais comuns.

Figura 28 – Tipos de notações de cardinalidades

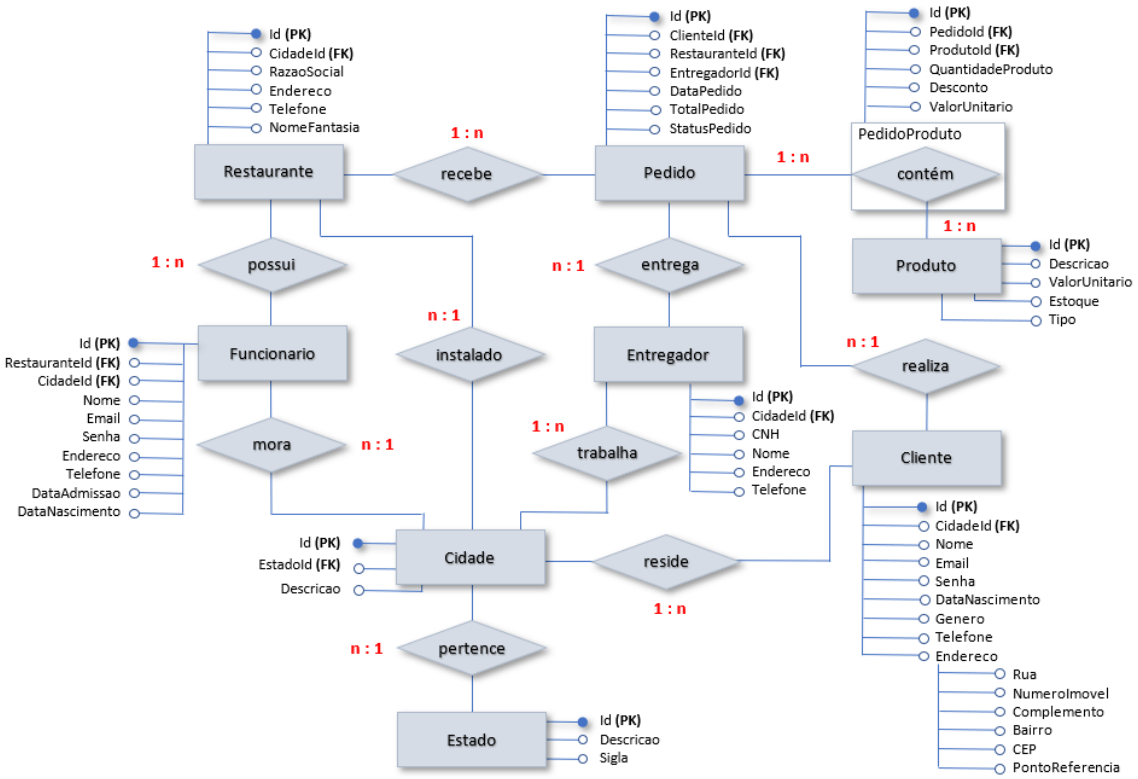
Notação – Cardinalidade			
Conectividade		Peter Chan	James Martin ('Crows foot', 'pé de galinha')
Notação simples	Mínima e máxima		
1:1	(1,1)	1 — 1	
1:n / n:1	(1,n)	1 — n	
n:n	(0,1) (0,n)	n — m	

Fonte: Albano; Albano, 2022.

5.5 Estudo de caso

Agora que você já conhece todos os tipos de cardinalidades, vamos aplicar no modelo do nosso estudo de caso.

Figura 29 – Inclusão de cardinalidades no modelo conceitual



Fonte: Albano; Albano, 2022.

Entendendo as cardinalidades**:**

- Estado/Cidade: uma cidade pertence a um estado e um estado pode possuir várias cidades em seu território. Cardinalidade resultante 1:n. A chave estrangeira será o ID da entidade Estado na entidade Cidade. Recorde que a chave estrangeira sempre será inserida na entidade cuja cardinalidade resultou em muitos (n).
- Funcionário/cidade: um funcionário mora em uma cidade e uma cidade pode ter nenhum ou vários funcionários morando. Cardinalidade resultante 1:n. A chave estrangeira será o ID da entidade Cidade na entidade Funcionário.
- Restaurante/Cidade: um restaurante está instalado em uma cidade e uma cidade pode ter nenhum ou vários restaurantes instalados. Cardinalidade resultante 1:n. A chave estrangeira será o ID da entidade Cidade na entidade Restaurante.
- Entregador/Cidade: um entregador trabalha em uma cidade e uma cidade pode ter nenhum ou vários entregadores trabalhando. Cardinalidade resultante 1:n. A chave estrangeira será o ID da entidade Cidade na entidade Entregador.
- Cliente/Cidade: um cliente reside em uma cidade e uma cidade pode ter nenhum ou vários clientes residindo. Cardinalidade resultante 1:n. A chave estrangeira será o ID da entidade Cidade na entidade Cliente.
- Funcionário/Restaurante: um funcionário trabalha em um restaurante e um restaurante pode ter um ou vários funcionários trabalhando. Cardinalidade resultante 1:n. A chave estrangeira será o ID da entidade Restaurante na entidade Funcionário.
- Restaurante/Pedido: um pedido é recebido por um restaurante e um restaurante recebe um ou vários pedidos. Cardinalidade resultante 1:n. A chave estrangeira será o ID da entidade Restaurante na entidade Pedido.
- Entregador/Pedido: um pedido será entregue por um entregador e um entregador faz a entrega de um ou vários pedidos. Cardinalidade 1:n. A chave estrangeira será o ID da entidade Entregador na entidade Pedido.
- Cliente/Pedido: um pedido será realizado por um cliente e um cliente realiza um ou vários pedidos. Cardinalidade 1:n. A chave estrangeira será o ID da entidade Cliente na entidade Pedido.

- Produto/Pedido: um pedido pode ter um ou vários produtos e um produto pode estar em nenhum ou vários pedidos. A cardinalidade resultante n:n. Aqui devemos observar que é necessário criar uma entidade associativa e, nesse caso, vamos chamar a entidade de PedidoProduto. Tal entidade possuirá como campos as chaves primárias das entidades Produto e Pedido, que assumem a função de chave estrangeira. Como chave primária usaremos a chave artificial ID. Além disso, outras informações também devem ser armazenadas nessa entidade, a quantidade do produto, o valor unitário do produto e o desconto do produto (aniversariantes).

Com a aplicação da cardinalidade, concluímos o desenvolvimento do modelo conceitual do nosso estudo de caso.

FINALIZANDO

Nesta etapa aprendemos sobre os conceitos fundamentais de Banco de Dados, conhecemos os modelos de armazenamento de dados, dos mais antigos (rede, hierárquico e orientação a objetos) até os atuais (relacional e No-SQL) e entendemos, ainda, como se dá a estrutura do Banco de Dados, com suas entidades, campos e relacionamentos.

Discutimos também sobre a importância do SGBD, que é o software que faz a comunicação entre o Banco de Dados, os usuários e as aplicações que acessam esses dados.

Aprendemos sobre as etapas do projeto de Banco de Dados e, juntos, desenvolvemos um modelo conceitual, baseado em um estudo de caso. O modelo desenvolvido é uma abstração de alto nível, ou seja, não apresenta a definição de detalhes técnicos.

Compreendemos o que é uma entidade, seus tipos, campos e os relacionamentos entre as entidades. Aprendemos sobre conceitos de chave primária, que diferencia uma instância das outras em uma entidade, e de chave estrangeira, que é o campo que faz a ligação entre as entidades.

Por fim, estudamos o que é cardinalidade, seus tipos (1:1, 1:n e n:n) e descobrimos que um relacionamento muitos para muitos (n:n) provoca o surgimento de uma nova entidade. Com isso, pudemos perceber a importância que a cardinalidade exerce na definição dos relacionamentos entre as entidades

do modelo, conceitos esses que também foram aplicados passo a passo no desenvolvimento do nosso estudo de caso.

Procure revisar os conceitos e os exemplos apresentados neste material para que você esteja preparado para os próximos passos que abrangerão a conversão do modelo conceitual para o modelo lógico e, subsequentemente, do lógico para o modelo físico.

REFERÊNCIAS

CHEN, Peter. **Modelagem de Dados: A abordagem entidade-relacionamento para projeto lógico**. São Paulo: Makron Books, 1990.