
APOSTILA DE BANCO DE DADOS

Parte 1

Profa. Rosana Massahud
CEFET MG - *Campus* Nepomuceno
Curso Técnico Redes de Computadores

NOTAS DE AULA

Com scripts para MySQL

Copyright ©2022
Todos os direitos reservados.
Versão 1.1

CURSO TÉCNICO EM REDES DE COMPUTADORES
CENTRO FEDERAL DE EDUCAÇÃO TECNOLÓGICA DE MINAS GERAIS
CAMPUS NEPOMUCENO

Caro estudante,

Bem-vindos à disciplina de Banco de Dados que tratará dos principais conceitos relacionados à área de banco de dados para aprofundar seus conhecimentos sobre modelagem, projeto, armazenamento e consulta de dados no curso Técnico em Redes de Computadores do Centro Federal de Educação Tecnológica de Minas Gerais – Campus Nepomuceno.

Para que seu estudo se torne proveitoso e prazeroso, esta disciplina foi organizada em Seções, com temas e subtemas que, por sua vez, são subdivididos em tópicos, atendendo aos objetivos do processo de ensino-aprendizagem.

A presente apostila (Parte 1) trata dos assuntos iniciais de banco de dados, como conceitos, fundamentos, modelos conceituais e de implementação, além de técnicas de normalização e breve introdução à SQL.

Vamos, então, iniciar nossas aulas? Bons estudos!

Rosana

Lista de Figuras

1.1	Configuração de um sistema de banco de dados simplificado	10
2.1	Entidades, atributos e relacionamento.	15
2.2	Diagrama esquemático para um banco de dados.	17
2.3	Arquitetura de três-esquemas.	18
2.4	Independência de dados	19
2.5	Várias visões sobre um banco de dados.	19
3.1	Processo para projeto de banco de dados.	23
3.2	Símbolos utilizados no DER e seus significados.	31
4.1	Notação EER para representar subclasses e especialização	40
4.2	Exemplos de generalização	41
4.3	Notação do diagrama EER para uma especialização definida por atributo	42
4.4	Exemplo de especialização sobreposta	43
4.5	Planilha de orçamento mensal	48
5.1	Exemplo de (parte) de banco de dados que armazena informações de aluno e disciplina	62
5.2	Exemplo de atributos e tuplas de uma relação ABC.	64
5.3	Notação do diagrama EER para uma especialização definida por atributo	67
5.4	Generalização. Generalizando CARRO e CAMINHAO na superclasse VEICULO . .	67
5.5	Notação de diagrama EER para uma especialização sobreposta (não disjunta)	68
5.6	Opções para mapeamento de especialização ou generalização. (a) Mapeando o esquema EER na Figura 5.3 ao usar a opção 8A. (b) Mapeando o esquema EER na Figura 5.4 ao usar a opção 8B. (c) Mapeando o esquema EER na Figura 5.3 ao usar a opção 8C. (d) Mapeando a Figura 5.5 ao usar a opção 8D com campos de tipo booleano Tipo_fabr e Tipo_compr	68
5.7	Esquema ER para o banco de dados MOVIMENTO_NAVIO	69
5.8	Diagrama ER para um esquema de banco de dados BANCO	70
5.9	Diagrama ER para um esquema de banco de dados COMPANHIA_AEREA	71
5.10	Diagrama ER para um esquema de banco de dados EMPRESA	72
5.11	Diagrama ER para um banco de dados de reservas de hotel	73
5.12	Entidade Relacionamento para um sistema de ‘fábrica de carros’	74
5.13	Diagrama EER para banco de dados de um complexo poliesportivo	76
5.14	Diagrama EER para banco de dados de um aeroporto	77
7.1	Modelo conceitual para o projeto Loja.	83
7.2	Modelo lógico para o projeto Loja.	84
7.3	Modificação no esquema do projeto Loja.	89
7.4	Modelo entidade relacionamento aluno-curso	95
7.5	Resultado do Script 38	100
7.6	Resultado do Script 40	101
8.1	Esquema de banco de dados relacional ‘Empresa’ simplificado	109
8.2	Exemplo de estado de banco de dados para o esquema relacional da Figura 8.1 . .	110
8.3	Exemplos de esquemas de tabelas sofrendo anomalias de atualização	110
8.4	Exemplos de estados para as tabelas da Figura 8.3	111
8.5	Projeto particularmente fraco para a relação FUNC_PROJ da Figura 8.3	112
8.6	Um estado da tabela ENSINA com uma possível dependência funcional TEXTO → DISCIPLINA.	113
8.7	Esquema de relação entre as Formas Normais.	113
8.8	Exemplo de esquema de relação na 1FN.	114

8.9	Normalização na 1FN. (a) Exemplo de relação que não está na 1FN. (b) Exemplo de estado da tabela DEPARTAMENTO	114
8.10	Possível estado da tabela Filmes1	117
8.11	Formulários de solicitação e ficha de livros	117
8.12	Formulários de solicitação e ficha de livros	118

Lista de Scripts

1	Criação do banco de dados	83
2	Criação do banco de dados, para MySQL	83
3	Criação do banco de dados loja	83
4	Criação do usuário adm	83
5	Privilégio concedido ao usuário ‘adm’	84
6	Comando para exclusão do esquema de banco de dados	84
7	Criação de Tabelas em Banco de Dados	85
8	Criação da tabela cliente no banco de dados loja	87
9	Criação da tabela cliente no banco de dados loja	87
10	Criação das tabelas produto e vendedor	87
11	Criação das tabelas venda, produto_venda e telefone	87
12	Comando para exclusão de tabelas	88
13	Renomeando o atributo nome da tabela cliente para nomeCli	88
14	Renomeando o atributo nome da tabela cliente para nomeCli	89
15	Adicionando atributos à tabela cliente	89
16	Removendo o atributo cidade da tabela cliente	89
17	Modificando o tipo de dados do atributo	89
18	Criação das tabelas cidade e estado.	89
19	Alteração da tabela cliente, acrescentando a coluna codCidade e a chave estrangeira.	90
20	Criação da chave primária e chave estrangeira após a criação da tabela.	90
21	Sintaxe básica do SELECT	92
22	Lista de todos os clientes	92
23	Lista de código e nome de todos os clientes	92
24	Lista de código e nome completo de todos os clientes	92
25	Consulta nome e sobrenome, de todos os clientes que são do estado do RJ	93
26	Consulta nome e sobrenome, de todos os clientes que são do estado do RJ ou do estado de MG	93
27	Consulta com condições ligadas por operador lógico	93
28	Lista de clientes com cidade ou data de nascimento não informadas.	93
29	Lista de clientes cujo primeiro nome é ‘Maria’	93
30	Exemplo de uso das funções UPPER e LOWER	94
31	Uso do ORDER BY	94
32	Listas ordenadas de modo crescente e descrescente	94
33	Exemplo de junção com operação de igualdade	95
34	Consulta de alunos e cursos usando INNER JOIN	96
35	Consulta de alunos e cursos usando OUTER JOIN	97
36	Consulta da quantidade de alunos por curso, usando INNER JOIN	99
37	Consulta da quantidade de alunos por curso, usando OUTER JOIN	99
38	Consulta com funções de agregação e agrupamento	99
39	Consulta com arredondamento	100
40	Consulta com agrupamento filtrado com o HAVING	100
41	Sintaxe básica de INSERT	101
42	INSERT com valores para todos os atributos da tabela	101
43	INSERT com valores de atributos específicos	101
44	Sintaxe básica do UPDATE	102
45	INSERT com valores de atributos específicos	102

46	Sintaxe básica do DELETE	102
47	Exemplo de DELETE na tabela Empregado, com condições	102

Sumário

Lista de Figuras	2
Lista de Scripts	4
1 Conceitos Gerais	9
1.1 O que é um Banco de Dados?	9
1.2 O que é um SGBD?	9
1.3 O que é um SBD?	10
1.4 Características da abordagem de banco de dados	10
1.5 Profissionais de Banco de Dados e suas atividades	11
1.6 Características da tecnologia de Banco de Dados	13
1.7 Quando não usar um SGBD	13
1.8 Perguntas de revisão	14
2 Arquitetura	15
2.1 Modelos de dados, esquemas e instâncias	15
2.2 Categorias de modelos de dados	15
2.3 Esquemas, instâncias e estado do banco de dados	16
2.4 Arquitetura de três esquemas e independência de dados	16
2.4.1 Independência dos dados	17
2.5 Linguagens do SGBD	18
2.6 Interfaces do SGBD	19
2.6.1 Interfaces baseadas em menus para os clientes web ou navegação	19
2.6.2 Interfaces baseadas em formulários	19
2.6.3 Interfaces gráficas para o usuário	19
2.6.4 Interface de linguagem natural	20
2.6.5 Interface para usuários parametrizáveis	20
2.6.6 Interfaces para o administrador do banco de dados	20
2.7 Arquiteturas centralizadas e cliente/servidor para os SGBDs	20
2.7.1 Arquitetura centralizada	20
2.7.2 Arquitetura cliente/servidor	20
2.8 Alguns exemplos de SGBDs	21
2.9 Perguntas de revisão	21
3 Modelo Entidade - Relacionamento - ER	22
3.1 Processo de projeto de Banco de Dados	22
3.2 Conceitos do MER - Entidade, Atributo e Relacionamento	24
3.2.1 Entidade	24
3.2.2 Atributos	24
3.2.3 Relacionamentos	25
3.2.4 Tipo de entidade fraca	30
3.3 Diagramas Entidade-Relacionamento (DER), Convenções de Nomenclatura e Decisões de Projeto	30
3.4 Notações para DER	30
3.5 Exercícios	31
3.6 Denominação dos construtores de Esquema	32
3.7 Decisões de projetos para o Projeto conceitual ER	33

3.8 Exercícios	33
4 O Modelo Entidade-Relacionamento Estendido - EER	39
4.1 Subclasse, Superclasse e Herança	39
4.2 Generalização e Especialização	39
4.3 Restrições sobre Especialização e Generalização	40
4.4 Exercícios	43
4.5 Estudos de caso	47
4.5.1 Estudo de caso 1: Sistema de Reservas em Hotel	47
4.5.2 Estudo de caso 2: Sistema de controle de orçamento	48
4.5.3 Estudo de caso 3: Sistema de Zoo	50
4.5.4 Estudo de caso 4: Sistema de Saúde	51
4.5.5 Estudo de caso 5: Salão de Beleza	52
4.5.6 Estudo de caso 6: Sistema de Imobiliária	53
4.5.7 Estudo de caso 7: Projeto de Bancos de Dados para Reclamações	54
4.5.8 Estudo de caso 8: Projeto de Bancos de Dados para Controle de acesso	56
4.5.9 Estudo de caso 9: Projeto de Banco de Dados para Controle de Frota	57
4.5.10 Estudo de caso 10: Sistema de Controle de Estoque	59
4.5.11 Estudo de caso 11: Sistema de Almoxarifado	60
5 Modelo Relacional	62
5.1 Introdução	62
5.2 Conceitos do modelo relacional	62
5.3 Propriedades de uma relação	63
5.4 Notação	64
5.5 Chave primária x Chave candidata x Chave estrangeira	64
5.6 Restrições do modelo relacional	65
5.6.1 Outros tipos de restrições	65
5.7 Projeto de um banco de dados relacional usando o mapeamento do ER para o relacional	65
5.7.1 Passos para o mapeamento ER - Relacional	66
5.7.2 Passos para o mapeamento EER - Relacional	66
5.8 Exercícios	69
6 Álgebra relacional	78
6.1 Introdução	78
6.2 Seleção	78
6.3 Projeção	78
6.4 Junção	79
6.5 União	80
6.6 Interseção	80
6.7 Diferença	80
6.8 Produto cartesiano	80
6.9 Divisão	81
7 SQL - Structured Query Language	82
7.1 Introdução	82
7.2 Linguagem de Definição de Dados (DDL)	82
7.2.1 Criação do esquema - CREATE SCHEMA	82

7.2.2	Criação de tabelas - CREATE TABLE	85
7.2.3	Excluindo tabelas - DROP TABLE	88
7.2.4	Alterações das definições de tabelas - ALTER TABLE	88
7.3	Linguagem de Manipulação de Dados (DML)	92
7.3.1	Consultas - SELECT	92
7.3.2	Inserções - INSERT	101
7.3.3	Atualizações - UPDATE	102
7.3.4	Remoção/exclusão de dados - DELETE	102
7.3.5	Restrições sobre as cláusulas DELETE e UPDATE	102
7.3.6	Outras cláusulas da SQL	103
7.4	Exercícios	104
8	Dependência Funcional e Normalização	108
8.1	Diretrizes de projeto informais para esquemas de relação	108
8.1.1	Semântica clara aos atributos nas relações	108
8.1.2	Informação redundante nas tuplas e anomalias de atualização	109
8.1.3	Valores NULL nas tuplas	110
8.1.4	Geração de tuplas falsas	111
8.2	Dependência Funcional	112
8.3	Normalização	113
8.3.1	Primeira Forma Normal (1FN)	114
8.3.2	Segunda Forma Normal (2FN)	115
8.3.3	Terceira Forma Normal (3FN)	115
8.3.4	Forma Normal Boyce-Codd	116
8.4	Exercícios	117
9	Referências Bibliográficas	121
Índice Remissivo		122

1 Conceitos Gerais

1.1 O que é um Banco de Dados?

Definição genérica: é uma coleção de dados relacionados. Com dados, queremos dizer fatos conhecidos que podem ser registrados e possuem significado implícito. Exemplo: considere os nomes, números de telefone e endereços das pessoas que você conhece. Essa coleção de dados relacionados, com um significado implícito, é um banco de dados.

Definição mais restrita (propriedades) :

- Representa algum aspecto do mundo real, às vezes chamado de minimundo ou de universo de discurso (UoD - Universe of Discourse)
- É uma coleção logicamente coerente de dados com algum significado inherente. Uma variedade aleatória de dados não pode ser corretamente chamada de banco de dados.
- É projetado, construído e povoado com um objetivo específico (para uma finalidade específica). Ele possui um grupo definido de usuários e algumas aplicações previamente concebidas nas quais esses usuários estão interessados.

Em outras palavras, um banco de dados tem alguma fonte da qual o dado é derivado, algum grau de interação com eventos no mundo real e um público que está ativamente interessado em seu conteúdo. Os usuários finais de um banco de dados podem realizar transações comerciais (por exemplo, um cliente compra uma câmera) ou eventos podem acontecer (por exemplo, uma funcionária tem um filho), fazendo com que a informação do banco de dados mude. Para que um banco de dados seja preciso e confiável o tempo todo, ele precisa ser um reflexo verdadeiro do minimundo que representa; portanto, as mudanças precisam ser refletidas no banco de dados o mais breve possível.

Um banco de dados pode ser gerado e mantido manualmente, ou pode ser computadorizado. Por exemplo, um caderno contendo informações de contatos, com telefones e endereços é um banco de dados que pode ser criado e mantido manualmente. Um banco de dados computadorizado pode ser criado e mantido por um grupo de programas de aplicação escritos especificamente para essa tarefa ou por um sistema gerenciador de banco de dados .

1.2 O que é um SGBD?

SGBD : Sistema Gerenciador de Banco de Dados

Definição 1: é uma coleção de programas que possibilita aos usuários criar e manter um banco de dados.

Definição 2: é um sistema de software de finalidade genérica que facilita o processo de definição¹, construção², manipulação³ e compartilhamento⁴ do banco de dados.

Um **programa de aplicação** acessa o banco de dados ao enviar consultas ou solicitações de dados ao **SGBD**. Uma **consulta** normalmente resulta na recuperação de alguns dados; uma **transação** pode fazer que alguns dados sejam lidos e outros, gravados no banco de dados.

¹Definição: a especificação dos tipos de dados, as estruturas e as restrições dos dados a serem armazenados.

²Construção: processo de armazenar dados em algum meio controlado pelo SGBD.

³Manipulação: permissão de consultas, atualizações e geração de relatórios. Inclui funções como consulta ao BD para recuperar dados específicos, atualizações do BD para refletir as mudanças no minimundo e geração de relatórios

⁴Compartilhamento de um banco de dados permite que diversos usuários e programas acessem-no simultaneamente

Outras funções importantes fornecidas pelo SGBD incluem *proteção* do banco de dados e sua *manutenção* por um longo período. A proteção inclui proteção do sistema contra defeitos (ou falhas) de hardware ou software e proteção de segurança contra acesso não autorizado ou malicioso.

Não é absolutamente necessário utilizar software de SGBD de uso geral para implementar um banco de dados computadorizado. Poderíamos escrever nosso próprio conjunto de programas para criar e manter o banco de dados, criando nosso próprio SGBD de *uso especial*. Em ambos os casos - se usarmos um SGBD de uso geral ou não -, em geral temos de implementar uma quantidade considerável de software complexo. De fato, a maioria dos SGBD's é constituída de sistemas de software muito complexos.

Para completar nossas definições iniciais, chamamos a união do banco de dados com o software de SGBD de **sistema de banco de dados**.

1.3 O que é um SBD?

SBD : Sistema de Banco de Dados

É um programa desenvolvido por usuários com funções específicas que utilizam programas responsáveis pela definição, construção, manipulação e compartilhamento de banco de dados, ou seja, SGBD , para acessarem os dados armazenados.

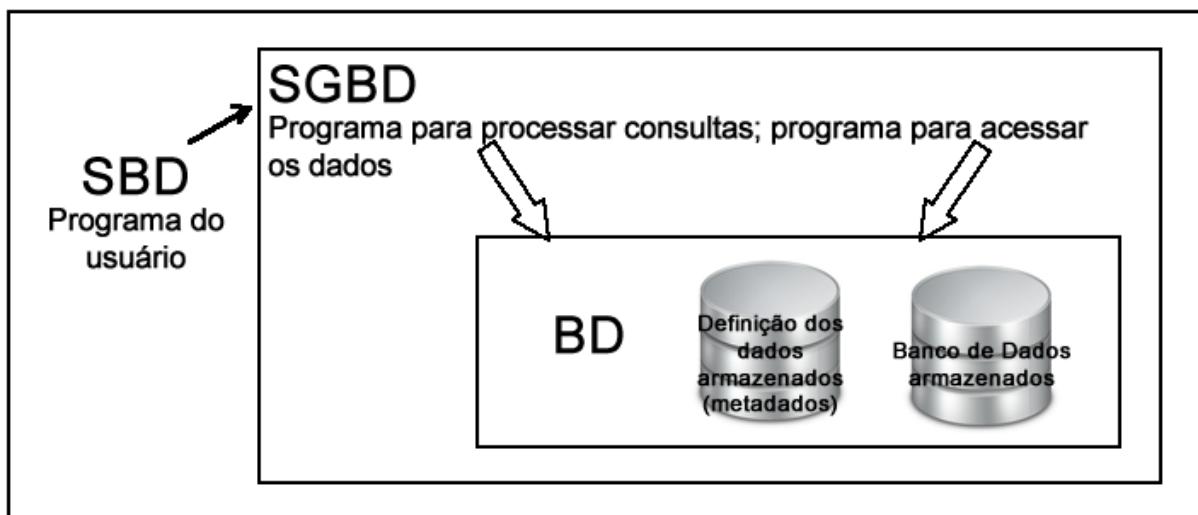


Figura 1.1: Configuração de um sistema de banco de dados simplificado

1.4 Características da abordagem de banco de dados

Diversas características distinguem a abordagem de banco de dados da abordagem muito mais antiga de programação com arquivos. No processamento de arquivo tradicional, cada usuário define e implementa os arquivos necessários para uma aplicação de software específica como parte da programação da aplicação. Por exemplo, um usuário, o departamento de registro acadêmico, pode manter arquivos sobre os alunos e suas notas. Os programas para imprimir o histórico escolar de um aluno e inserir novas notas são implementados como parte da aplicação. Um segundo usuário , o departamento de finanças, pode registrar as mensalidades dos alunos e seus pagamentos. Embora ambos os usuários estejam interessados em dados sobre alunos, cada um mantém arquivos separados - e programas para manipular esses arquivos -, pois cada usuário requer dados não disponíveis

nos arquivos do outro. Essa redundância na definição e no armazenamento de dados resulta em desperdício no espaço de armazenamento e em esforços redundantes para manter os dados comuns atualizados.

Na abordagem de banco de dados, um único repositório mantém dados que são definidos uma vez e depois acessados por vários usuários. Nos sistemas de arquivo, cada aplicação é livre para nomear os elementos de dados independentemente. Ao contrário, em um banco de dados, os nomes ou rótulos de dados são definidos uma vez, e usados repetidamente por consultas, transações e aplicações.

As principais características da abordagem de banco de dados versus a abordagem de processamento de arquivo são as seguintes:

- Natureza de autodescrição de um sistema de banco de dados (**metadados**).
- Isolamento entre programas e dados, e abstração de dados . (**Independência de dados do programa** e **Independência da operação do programa**). A característica que permite a independência de dados do programa e a independência da operação do programa é chamada de **abstração de dados**. Um SGBD oferece aos usuários uma **representação conceitual** de dados que não inclui muitos dos detalhes de como os dados são armazenados ou como as operações são implementadas. Um **modelo de dados** é um tipo de abstração de dados usado para oferecer essa representação conceitual .
- Suporte de múltiplas visões dos dados .
- Compartilhamento de dados e processamento de transação multiusuário . O SGBD precisa incluir um software de controle de concorrência para garantir que vários usuários tentando atualizar o mesmo dado façam isso de uma maneira controlada, de modo que o resultado dessas atualizações seja correto. Por exemplo, quando vários agentes de viagem tentam reservar um assento em um voo, o SGBD precisa garantir que cada assento só possa ser acessado por um agente cada vez para que seja atribuído a um único passageiro. Esses tipos de aplicações geralmente são chamados de aplicações de **processamento de transação online (OLTP - Online Transaction Processing)**.

O conceito de transação é fundamental para muitas aplicações de banco de dados. Uma transação é um programa em execução ou processo que inclui um ou mais acessos ao banco de dados, como a leitura ou atualização de seus registros. Uma transação executa um acesso logicamente correto a um banco de dados quando ela é executada de forma completa e sem interferência de outras transações. O SGBD precisa impor várias propriedades da transação . A propriedade de isolamento garante que cada transação pareça executar isoladamente das demais, embora centenas de transações possam estar executando concurrentemente. A propriedade de atomicidade garante que todas as operações em uma transação sejam executadas ou que nenhuma seja.

1.5 Profissionais de Banco de Dados e suas atividades

- Administrador de Banco de Dados (DBA - database administrator) De um modo geral, sua função é cuidar da saúde do banco de dados . Tanto em manter sua disponibilidade quanto em otimizá-lo. Em um ambiente de BD, o recurso principal é o próprio banco de dados, e o recurso secundário é o SGBD e os softwares relacionados. A administração desses recursos é de responsabilidade do DBA. Ele é responsável por:
 - Administrar o BD e o SGBD

- Automatizar o acesso ao BD
- Coordenar e monitorar o uso do BD
- Requisitar recursos de hardware e software
- Resolver questões de violação de segurança
- Agilizar o tempo de resposta

Em grandes organizações, o DBA é auxiliado por uma equipe.

- **Projetista de banco de dados**

São os responsáveis por identificar os dados a serem armazenados e escolher estruturas apropriadas para representar e armazenar esses dados. Essas tarefas são realizadas antes que o BD esteja realmente implementado e populado com dados. Os projetistas são responsáveis por:

- Identificar os dados a serem armazenados
- Escolher estruturas adequadas
- Conversar com os usuários para entender suas necessidades
- Construir um modelo para atender essas necessidades
- Desenvolver visões dos dados para atender usuários

- **Analistas de Dados**

Coletar, compilar, analisar e interpretar os dados . Estes dados podem ser usados para promover um negócio em termos de desenvolvimento de novas estratégias de marketing e, de fato, utilizado para finalmente colher lucros para a empresa.

- **Desenvolvedor / Programador de SQL**

Tem como função programar/desenvolver scripts SQL para atender as regras de negócio do banco. Trabalham geralmente com as linguagens PL/SQL (Oracle) ou T/SQL (SQL Server), ou a linguagem procedural associada ao SGBD específico, como MySQL, PostgreSQL, etc.

- **Analista de Sistemas/ Suporte**

- Analisa e desenvolve sistemas, mapeia processos, faz a modelagem de dados e levanta os requisitos para implementar esses programas de acordo com os objetivos e as regras de negócio da empresa.
- Dependendo da empresa um analista pode desenvolver as funções de DBA , analista de dados , programador SQL e até analista de BI.
- Há uma ligação forte entre sistemas ERP e banco de dados . Uma das atividades do analista de sistemas /suporte é de entender os processos do ERP e a estrutura de dados da base que o sistema foi implementado, para assim gerar relatórios e otimizar os fluxos do sistema.

- **Analista de Business Intelligence (BI)**

Montar os modelos de negócio, levantamento de requisitos , criação de cubos de dados e templates de relatórios que precisarão abordar todos os cenários previamente definidos, após coleta, organização e análise das informações de mercado que dão suporte à gestão do negócio.

- Cientista de Dados

Especialista com habilidade para analisar (grande volume de dados – BIG DATA) e interpretar informações de valor e apoiar na tomada de decisão dos negócios.

- Usuários finais

São pessoas cujas funções exigem acesso ao banco de dados para consultas, atualizações e geração de relatórios. O banco de dados existe para atender os usuários finais. Existem várias categorias de usuários finais:

- Casuais: ocasionalmente acessam o BD
- Leigos: executam consultas previamente estabelecidas
- Sofisticados: conhecedoras das facilidades do SGBD
- Individuais ou isolados: Mantém BDs pessoais usando pacotes de programas prontos, que oferecem interfaces de fácil utilização.

Os salários dos profissionais podem variar de acordo com o nível profissional (estagiário, júnior, pleno, sênior, master/especialista), porte da empresa (pequena, média, grande) e também pela localização da empresa; por exemplo, em alguns estados brasileiros paga-se melhor que em outros, assim como alguns países a profissão é mais valorizada que em outros.

1.6 Características da tecnologia de Banco de Dados

Controle de redundância: evita a duplicação de esforços; não permite inconsistências e economiza espaços de armazenamento;

Compartilhamento de dados : fornece controle de concorrência para garantir atualizações simultâneas;

Restrições de acesso: evita o acesso de pessoas não autorizadas;

Múltiplas interfaces: fornece interface específica para cada tipo de usuário ;

Relacionamentos complexos: representa vários relacionamentos e facilita a recuperação e a atualização dos dados;

Regras de integridade : A maioria das aplicações de banco de dados possui certas restrições de integridade que devem ser mantidas para os dados . São as regras de negócio. Um SGBD deve oferecer capacidade para definir e impor tais restrições;

Backup e restore (recuperação): Um SGBD precisa oferecer recursos para recuperar-se de falhas de hardware ou software.

1.7 Quando não usar um SGBD

Apesar das vantagens de usar um SGBD , existem situações em que esse sistema pode envolver custos adicionais desnecessários, que não aconteceriam no processamento de arquivos tradicional. Os custos adicionais do uso de um SGBD devem-se aos seguintes fatores:

- Alto investimento inicial em hardware, software e treinamento.
- A generalidade que um SGBD oferece para a definição e o processamento de dados.

- Esforço adicional para oferecer funções de segurança, controle de concorrência, recuperação e integridade.

Portanto, pode ser mais desejável usar arquivos comuns sob as seguintes circunstâncias:

- Aplicações de banco de dados simples e bem definidas, para as quais não se espera muitas mudanças.
- Requisitos rigorosos, de tempo real, para alguns programas de aplicação, que podem não ser atendidos devido as operações extras executadas pelo SGBD .
- Sistemas embarcados com capacidade de armazenamento limitada, onde um SGBD de uso gerao não seria apropriado.
- Nenhum acesso de múltiplos usuários aos dados .

1.8 Perguntas de revisão

1. Defina os seguintes termos: dados, banco de dados, SGBD, sistema de banco de dados, metadados, independência entre dados e programas, visão do usuário, DBA, usuário final, e transação.
2. Quais os quatro tipos principais de ações que envolvem banco de dados? Discuta cada tipo rapidamente.
3. Discuta as principais características da abordagem de banco de dados e como ela difere dos sistemas de arquivo tradicionais.
4. Quais as responsabilidades do DBA e dos projetistas de banco de dados?
5. Discuta as capacidades que devem ser fornecidas por um SGBD.

2 Arquitetura

2.1 Modelos de dados, esquemas e instâncias

Uma das principais características da abordagem de banco de dados é possibilitar a **abstração de dados**, de modo que diferentes usuários possam percebê-los em seu nível de detalhe preferido. Um **modelo de dados** é um conjunto de ferramentas conceituais para a descrição de dados, relacionamento de dados, semântica de dados e regras de consistência, ou seja, é um conjunto de conceitos que podem ser utilizados para descrever a estrutura de um banco de dados. Um modelo de dados, oferece os meios necessários para alcançarmos a abstração dos dados.

São operações básicas em um modelo de dados: inclusão, exclusão, modificação e recuperação dos dados.

Para construir um modelo de dados, usa-se uma linguagem de modelagem de dados que pode ser gráfica ou textual. Cada representação do modelo de dados recebe a denominação de **esquema de banco de dados**.

2.2 Categorias de modelos de dados

De acordo com a intenção do modelador, um banco de dados pode ser modelado de vários níveis de abstração. Um modelo que servirá para explicar a um usuário (leigo) não conterá detalhes sobre a representação em meio físico (mais abstrato), por outro lado, se o modelo servir para um especialista em banco de dados, ele conterá mais detalhes (menos abstrato).

Assim sendo, pode-se categorizar os modelos de dados baseando-se nos tipos de conceitos que fornecem para descrever a estrutura de base de dados.

- Modelo de dados conceitual (alto nível): fornece conceitos próximos ao entendimento (concepção) do usuário;
- Modelo de dados físico (baixo nível) : fornece conceitos que descrevem detalhes de como os dados são armazenados
- Modelo de dados de implementação : fornece conceitos de relativo entendimento ao usuário e próximo de como os dados são armazenados.

Os modelos de dados conceituais utilizam conceitos como *entidade*, *atributo* e *relacionamento*. Uma **entidade** representa um objeto ou conceito do mundo real, como um funcionário ou um projeto do minimundo que será descrito no banco de dados ; um **atributo** representa alguma propriedade de interesse que descreve melhor uma entidade, como nome ou salário do funcionário; um **relacionamento** entre duas ou mais entidades, representa uma associação entre elas, por exemplo, o relacionamento *trabalha* entre um *funcionário* e um *projeto*. O **modelo Entidade-Relacionamento** é um modelo de dados conceitual popular de alto nível.



Figura 2.1: Entidades, atributos e relacionamento.

Os modelos de dados de implementação (ou representativos) são frequentemente utilizados nos SGBDs comerciais tradicionais. Estes incluem o (ainda) amplamente utilizado **modelo de dados relacional**. Os modelos de dados representativos mostram os dados usando estruturas de registro e, portanto, às vezes são denominados **modelos de dados baseados em registro**.

Os modelos de dados físicos descrevem o armazenamento dos dados como arquivos no computador, com informações como formatos de registro, ordenações de registro e caminhos de acesso (uma estrutura que facilita a busca da informação).

2.3 Esquemas, instâncias e estado do banco de dados

Em qualquer modelo de dados, é importante distinguir entre a *descrição do banco de dados* e o *próprio banco de dados*. A descrição de um banco de dados é chamada **esquema**⁵, que é especificado durante o projeto de banco de dados e não se espera que mude com frequência. A maioria dos modelos de dados tem certas convenções para representar esquemas como diagramas. A representação de um esquema é chamada de **diagrama de esquema**.

Os dados de um banco de dados podem mudar com relativa frequência. Os dados de um banco de dados em um determinado momento do tempo são chamados estados ou instante do banco de dados; também são chamados de conjunto atual de ocorrências ou instância do banco de dados⁶

A distinção entre esquema e estado de banco de dados é muito importante. Quando definimos um novo banco de dados, especificamos seu esquema apenas para o SGBD. Nesse ponto, o estado do banco de dados correspondente é o *estado vazio*, sem dados. Obtemos o *estado inicial* do banco de dados quando ele é populado ou carregado com os dados iniciais. Daí em diante, toda vez que uma operação de atualização é aplicada, obtemos outro estado do banco de dados. Em qualquer ponto no tempo, o banco de dados tem um *estado atual*. O SGBD é parcialmente responsável por garantir que todo estado do banco de dados seja um **estado válido**, ou seja, um estado que satisfaça a estrutura e as restrições especificadas no esquema.

O SGBD armazena as descrições das construções e restrições do esquema - **metadados** - no catálogo do SGBD, de modo que o software do SGBD possa recorrer ao esquema sempre que precisar.

Embora o esquema não deva mudar com frequência, não é raro que as mudanças, ocasionalmente, precisem ser aplicadas ao esquema, à medida que os requisitos da aplicação mudam. Isso é conhecido como **evolução do esquema**. A maioria dos SGBDs modernos possui algumas operações para evolução de esquema que podem ser aplicadas enquanto o banco de dados está em funcionamento.

2.4 Arquitetura de três esquemas e independência de dados

Vimos anteriormente que três das quatro características importantes da abordagem de banco de dados são: (1) uso de um catálogo para armazenar a descrição (esquema) do banco de dados de modo a torná-lo autodescritivo, (2) isolamento de programas e dados (independência entre dados do programa e operação do programa) e (3) suporte para múltiplas visões do usuário. A conhecida **arquitetura de três esquemas** foi proposta para ajudar a alcançar e visualizar essas características. Veja na Figura 2.3. Nessa arquitetura os esquemas podem ser definidos nos três níveis a seguir:

Nível externo: possui esquema externo. Descreve uma visão do BD para grupos de usuários. Usa o modelo de dados conceitual (MDC).

⁵Esquema: é a descrição do banco de dados.

⁶Instância : os dados no banco de dados, em um determinado momento. Também chamado de estado do banco de dados ou instantâneo (*snapshot*), ou conjunto atual de ocorrências.

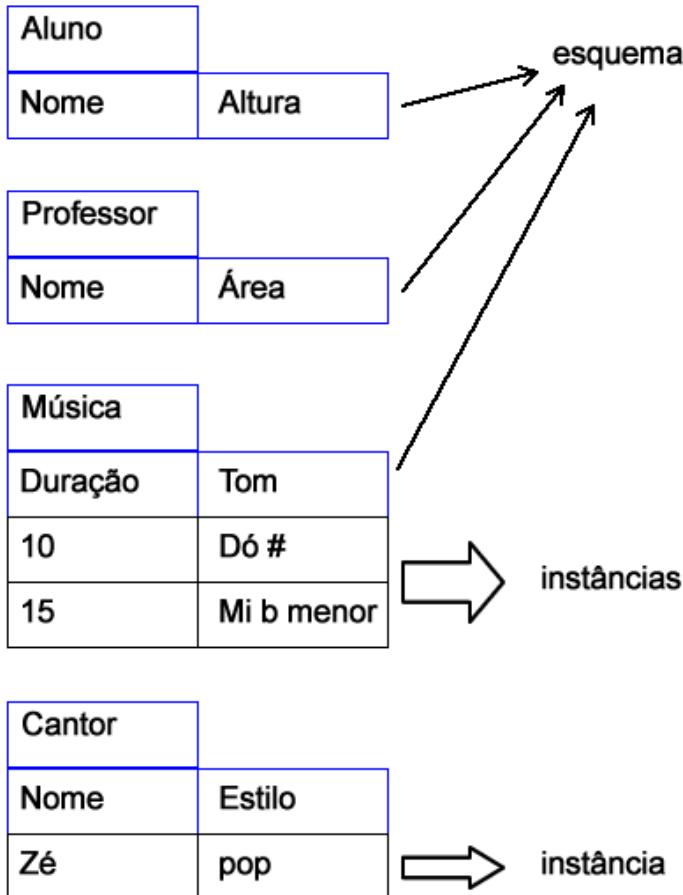


Figura 2.2: Diagrama esquemático para um banco de dados.

Nível conceitual: Possui esquema conceitual. Descreve a estrutura do BD, omitindo detalhes de armazenagem física e descrevendo entidades, relacionamentos e restrições. Usa o modelo de dados de implementação.

Nível interno: Possui esquema interno. Descreve a estrutura de armazenamento físico. Usa modelo de dados físico.

A arquitetura de três esquemas é uma ferramenta com a qual o usuário pode visualizar os níveis de esquema em um sistema de banco de dados. A maioria dos SGBDs não separa os três níveis completa e explicitamente, mas dá suporte a eles de alguma forma.

2.4.1 Independência dos dados

Os conceitos de independência de dados podem ser definidos como a capacidade de alterar o esquema de um nível sem ter que alterar o esquema no nível anterior.

Independência de dados lógica: É a capacidade de modificar o esquema conceitual sem alterar o esquema externo.

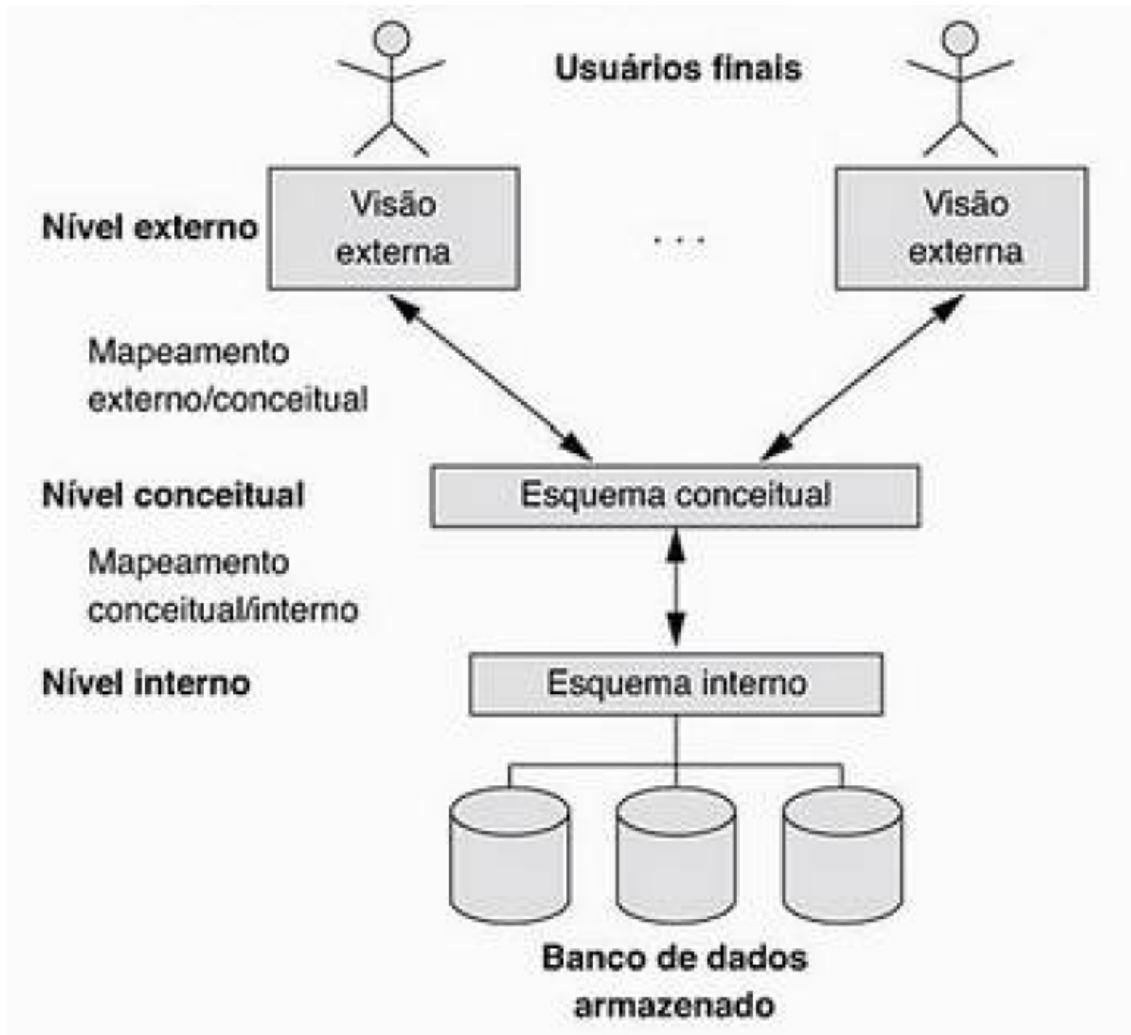


Figura 2.3: Arquitetura de três-esquemas.

Independência física de dados: É a capacidade de alterar o esquema interno sem alterar o esquema conceitual .

A independência dos dados é atingida quando o esquema de um nível é alterado e o esquema do nível superior não é atingido. Desta forma, a Independência Lógica de Dados , ILD, é mais difícil de ser alcançada que a Independência Física de Dados , IFD, pois os esquemas externos são mais dependentes da estrutura lógica de dados que de seu acesso.

2.5 Linguagens do SGBD

- DDL (Data Definition Language) É utilizada pelo DBA para definir os esquemas .
- SDL (Storage Definition Language) Utilizada para especificar o esquema interno .
- VDL (View Definition Language) Utilizada para especificar visões aos usuários e mapeamento para o esquema conceitual (Subconjuntos da tabela). Veja Figura 2.5.

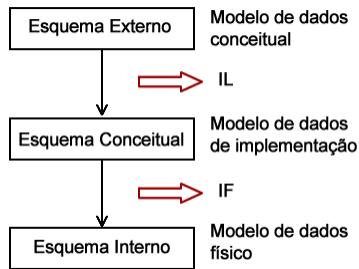


Figura 2.4: Independência de dados .

- DML (data Manipulation Language) Envolve as operações de inserção , remoção , alteração e recuperação .

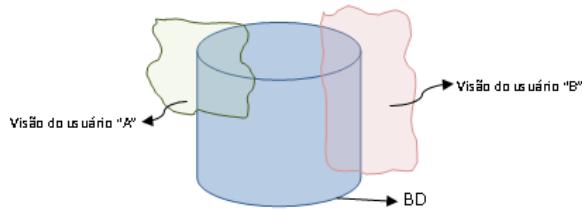


Figura 2.5: Várias visões sobre um banco de dados.

2.6 Interfaces do SGBD

As interfaces do SGBD podem ser subdivididas em:

2.6.1 Interfaces baseadas em menus para os clientes web ou navegação

Esse tipo de interface apresenta ao usuário listas de opções, os menus, que o guiam durante a formulação de uma pesquisa. Assim, não há necessidade de memorizar comando específicos ou sintaxes de linguagem para realizar uma consulta. Esse tipo de menu geralmente é utilizado nas interfaces para navegação e permitindo que o usuário pesquise o conteúdo de um banco de dados de uma forma exploratória e não estruturada.

2.6.2 Interfaces baseadas em formulários

Exibe um formulário para cada usuário. Não há necessidade de preencher todos os campos do formulário para realizar uma consulta . Normalmente, os formulários são projetados e programados para os usuários iniciantes como interfaces para transações customizadas. Alguns SGBDs contêm funcionalidades que permitem que o usuário final construa, interativamente, um formulário na tela.

2.6.3 Interfaces gráficas para o usuário

Uma interface gráfica para o usuário exibe um esquema para o usuário em um formulário diagramático e a consulta pode ser especificada manipulando o diagrama . Em alguns casos, as interfaces gráficas para o usuário utilizam menus e formulários

2.6.4 Interface de linguagem natural

Esse tipo de interface aceita solicitações escritas em inglês ou em outros idiomas e tentam entendê-las. Se a interface conseguir interpretar as solicitações feitas pelo usuário, uma consulta de alto nível é gerada e submetida ao processamento pelo SGDB. Um diálogo é então iniciado com o usuário para esclarecer a solicitação.

2.6.5 Interface para usuários parametrizáveis

Em geral, uma pequena série de comandos adaptados é disponibilizada com o objetivo de minimizar o número de teclas para cada solicitação. As teclas de funções de um terminal, por exemplo, podem ser programadas para iniciar os vários comandos, o que permite ao usuário parametrizável trabalhar com um numero mínimo delas.

2.6.6 Interfaces para o administrador do banco de dados

Alguns comandos de dados privilegiados são disponibilizados pela maioria dos sistemas de bancos de dados e que apenas administradores de bancos de dados podem utilizá-los . Esses comandos incluem criação de contas, sistema de ajuste de parâmetros, autorizações para criações de contas, mudança de esquemas e reorganização de estruturas de armazenamento do banco de dados .

2.7 Arquiteturas centralizadas e cliente/servidor para os SGBDs

2.7.1 Arquitetura centralizada

Na arquitetura centralizada, existe um computador com grande capacidade de processamento, o qual é o hospedeiro do SGBD e emuladores para os vários aplicativos . Esta arquitetura tem como principal vantagem a de permitir que muitos usuários manipulem grande volume de dados. Sua principal desvantagem está no seu alto custo, pois exige ambiente especial para mainframes e soluções centralizadas.

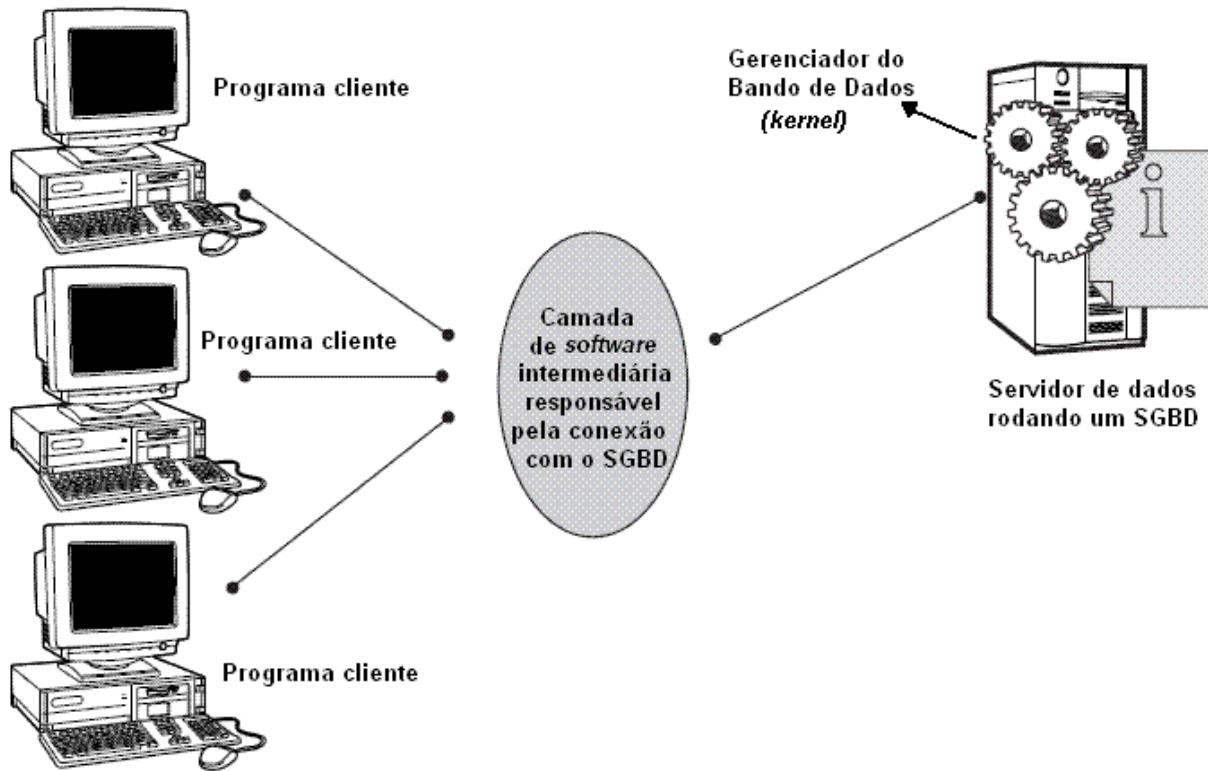


2.7.2 Arquitetura cliente/servidor

Na arquitetura Cliente/Servidor, o cliente (*front-end*) executa as tarefas do aplicativo, ou seja, fornece a interface do usuário (tela, e processamento de entrada e saída). O servidor (*back-end*) executa as consultas no SGBD e retorna os resultados ao cliente.

Apesar de ser uma arquitetura bastante popular, são necessárias soluções sofisticadas de software que possibilitem: o tratamento de transações , as confirmações de transações (*commits*) , desfazer transações (*rollbacks*) , linguagens de consultas (*stored procedures*) e gatilhos (*triggers*) .

A principal vantagem desta arquitetura é a divisão do processamento entre dois sistemas, o que reduz o tráfego de dados na rede.



2.8 Alguns exemplos de SGBDs

- IBM Informix
- PostgreSQL
- Firebird
- IBM DB2
- MySQL
- Oracle
- SQL-Server
- SQLite
- Sybase ...

2.9 Perguntas de revisão

1. Defina os seguintes termos: modelo de dados, esquema de banco de dados, estado de banco de dados, independência de dados, DDL, DML, VDL.
2. Qual é a diferença entre um esquema de banco de dados e um estado de banco de dados?
3. Qual é a diferença entre a independência lógica e a independência física dos dados? Qual é a mais difícil de se alcançar? Por quê?

3 Modelo Entidade - Relacionamento - ER

O Modelo Entidade Relacionamento (MER) é um modelo de dados conceitual. Assim, os conceitos do MER foram projetados para serem compreensíveis aos usuários.

O MER é usado principalmente durante o processo de projeto de Banco de Dados.

Este modelo tem por base a percepção de que o mundo real é formado de entidades e relacionamentos. O MER é um dos modelos de maior capacidade semântica. É representado graficamente pelo Diagrama Entidade-Relacionamento (DER).

3.1 Processo de projeto de Banco de Dados

O primeiro passo para o projeto de um banco de dados é o levantamento e análise de requisitos. Durante essa etapa, o projetista entrevista o possível usuário do banco de dados para entender e documentar seus requisitos de dados. O resultado dessa etapa é o registro conciso dos requisitos do usuário. Esses requisitos deveriam ser especificados em um formulário, da forma mais detalhada e completa possível. Em paralelo à especificação dos requisitos de dados, é útil definir os requisitos funcionais conhecidos da aplicação. Esses requisitos consistem em operações (ou transações) definidas pelo usuário que serão empregadas no banco de dados, incluindo as recuperações e atualizações. No projeto de software, é comum o uso de diagramas de fluxo de dados, diagramas de sequência, cenários e outras técnicas para a especificação de requisitos funcionais.

Uma vez que todos os requisitos tenham sido levantados e analisados, o próximo passo é criar um esquema conceitual para o banco de dados, utilizando um modelo de dados conceitual de alto nível. Essa fase é chamada projeto conceitual. O esquema conceitual é uma descrição concisa dos requisitos de dados dos usuários e inclui descrições detalhadas de tipos entidade, relacionamentos e restrições – são expressos usando os conceitos fornecidos pelo modelo de dados de alto nível. Como esses conceitos não incluem detalhes de implementação, eles são, normalmente, mais fáceis de entender e podem ser empregados na comunicação com os usuários não-técnicos. O esquema conceitual de alto nível também pode ser usado como uma referência para assegurar que todos os requisitos de dados do usuário sejam atendidos e não entrem em conflito. Essa abordagem permite que os projetistas de banco de dados se concentrem na especificação das propriedades do dado, sem se preocupar com os detalhes de armazenamento. Consequentemente, é mais fácil apresentarem um bom projeto conceitual do banco de dados.

Durante ou após o projeto do esquema conceitual, as operações básicas do modelo de dados podem ser usadas para especificar as operações de alto nível do usuário, identificadas durante a análise funcional. Servem também para confirmar que o esquema conceitual reúne todos os requisitos funcionais identificados. As modificações no esquema conceitual podem ser feitas se alguns requisitos funcionais não puderem ser especificados usando-se o esquema inicial.

A próxima etapa no projeto do banco de dados é a implementação real do banco de dados utilizando um SGBD comercial. A maioria dos SGBDs comerciais atuais usa um modelo de dados de implementação – como o relacional ou o modelo de banco de dados objeto-relacional – de forma que o esquema conceitual seja transformado de um modelo de dados de alto nível em um modelo de dados de implementação. Essa fase é conhecida como projeto lógico ou mapeamento do modelo de dados, e o seu resultado é um esquema do banco de dados no modelo de dados de implementação do SGBD.

O último passo é a fase do projeto físico, durante a qual são definidas as estruturas de armazenamento interno, índices, caminhos de acesso e organizações de arquivo para os arquivos do banco de dados. Em paralelo a essas atividades são projetados e implementados os programas de aplicação, como transações do banco de dados correspondentes às especificações de transação de alto nível.

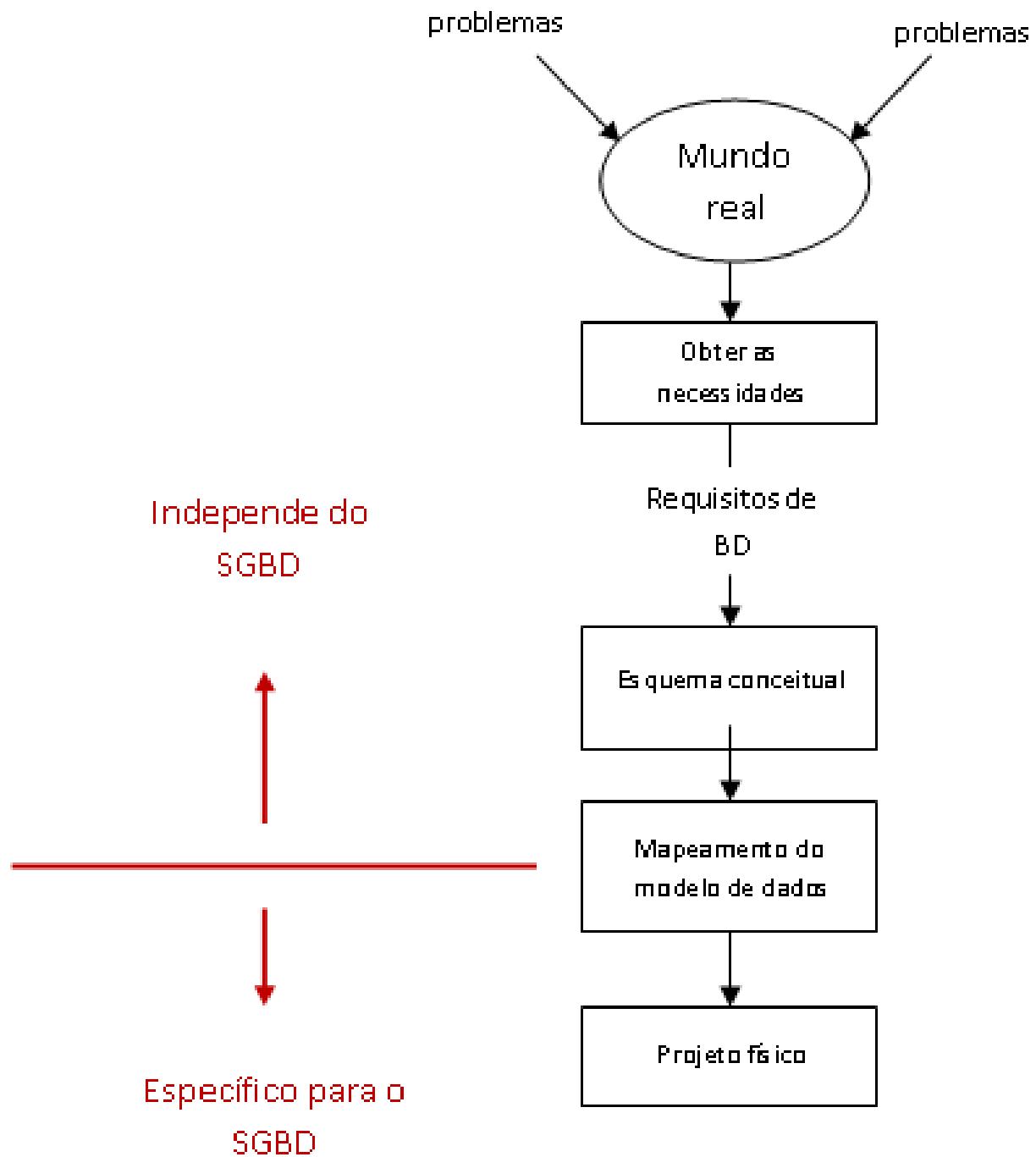


Figura 3.1: Processo para projeto de banco de dados.

3.2 Conceitos do MER - Entidade, Atributo e Relacionamento

3.2.1 Entidade

Identifica o objeto de interesse do sistema e tem “vida” própria, ou seja, a representação abstrata de um objeto do mundo real sobre o qual desejamos guardar informações.

Exemplo: Clientes, Fornecedores, Alunos, Funcionários, Departamentos, etc.

Não são entidades:

- Entidade com apenas 01 elemento;
- Operações do sistema;
- Saídas do sistema;
- Pessoas que realizam trabalhos (usuários do sistema);
- Cargos de direção

Um tipo de entidade é um conjunto de todas as entidades que compartilham as mesmas propriedades (atributos).

Contudo, cada entidade possui valores próprios para cada atributo.

Por exemplo:

Os alunos de uma sala constituem um tipo de entidade - Aluno.

Os livros de uma sala constituem um tipo de entidade - Livro.

O conjunto de todos os valores de um atributo é chamado de domínio do atributo.

Aluno	Nome - Todas as Strings
Livro	ISBN - Números inteiros
Aluno	Peso - Todos os números reais entre 80 e 160
Livro	Preço - Todos os números reais ≤ 100

Num diagrama entidade-relacionamento, representaremos as entidades com um retângulo contendo o seu nome.

3.2.2 Atributos

São as informações que desejamos guardar sobre a instância de entidade.

Simples x Compostos

- **Simples:** São atributos não divisíveis.
- **Compostos:** Podem ser subdivididos em outros atributos.

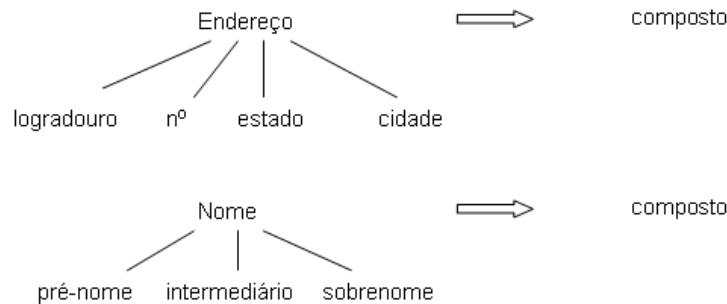
Exemplos: Nome, peso, altura - simples

Univalorado x Multivalorado

- **Univalorado:** possui um único valor.
- **Multivalorado:** possui mais de um valor

Exemplos: Nome, ISBN - univalorado Telefone - multivalorado

Primário (primitivo) x Derivado



- **Primitivo:** quando não é possível obter um valor através da derivação de outro atributo
- **Derivado:** seu valor é obtido através de outro atributo

Exemplos: idade - derivado (data de nascimento) data de nascimento - primitivo

Atributo nulo:

- Não possui valor. Por exemplo, no tipo de entidade aluno, o atributo telefone para a entidade João não tem valor, pois realmente a entidade não tem telefone.
- Omissão. Por exemplo, a entidade João não informou o seu telefone
- Não se aplica. Por exemplo, a entidade João não possui valor para o atributo complemento pois mora em uma casa.

Atributo chave:

Este atributo possui valor distinto para todas as entidades de um tipo de entidade. Assim, esta restrição proíbe que duas entidades tenham o mesmo valor para este atributo.

Num diagrama entidade-relacionamento, representaremos os atributos de uma entidade com um oval ligado à entidade. De acordo com cada classificação do atributo teremos um representação distinta. Veja na seção “Notações para DER”.

3.2.3 Relacionamentos

Um relacionamento é uma associação entre duas ou mais entidades.

Exemplo: O João está matriculado na disciplina de Banco de Dados. Onde:

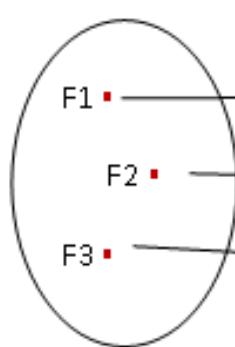
- “João”: Elemento do conjunto de valores do atributo Nome do aluno da entidade Aluno;
- “Banco de Dados”: Elemento do conjunto de valores do atributo Nome da disciplina da entidade Disciplina;
- “matriculado”: Ligação existente entre um aluno e uma disciplina.

O grau de um tipo de relacionamento indica o número de tipos de entidades participantes.



Veja outros exemplos:

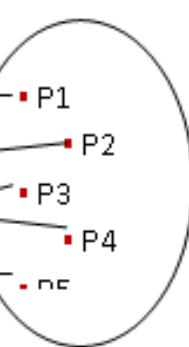
Fornecedor



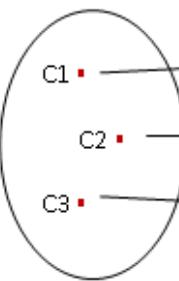
fornecce



Peça



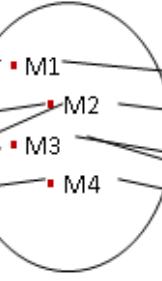
Cantor



canta



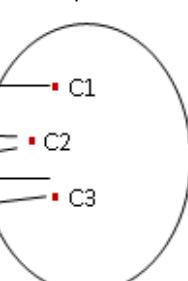
Música

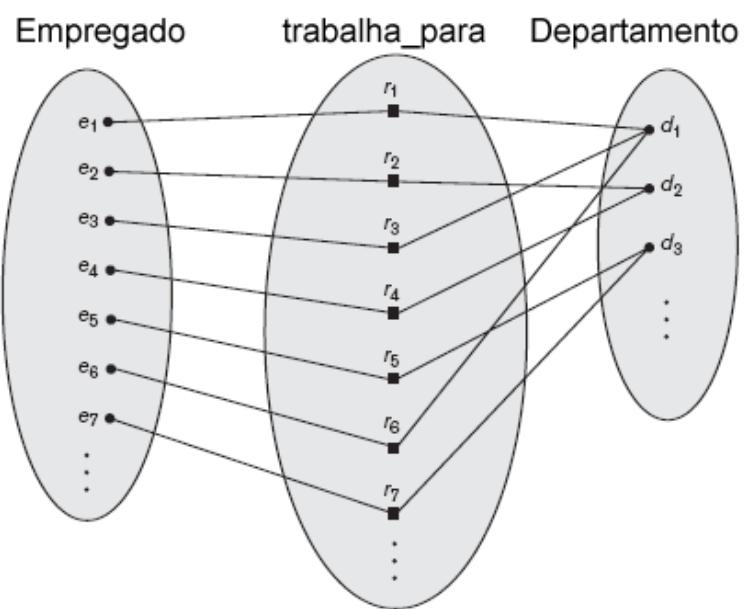
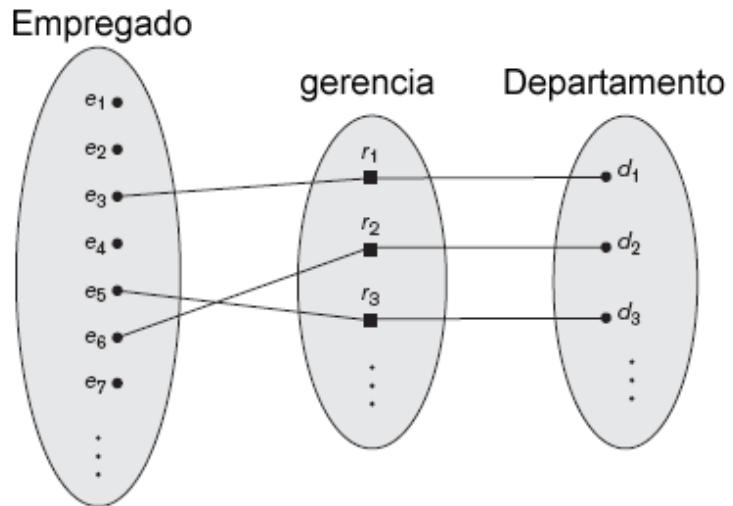


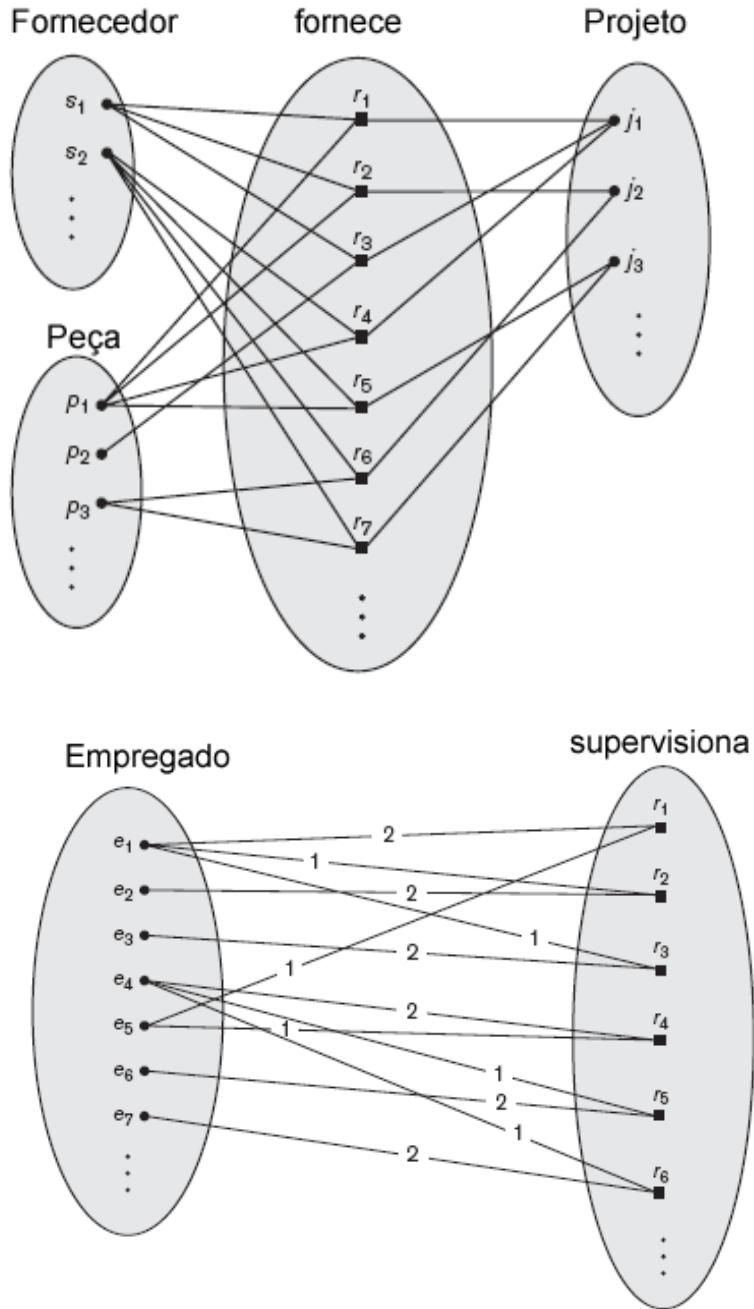
escreve



Compositor







Num diagrama entidade-relacionamento, representaremos um tipo de relacionamento com um losango ligando as entidades envolvidas.

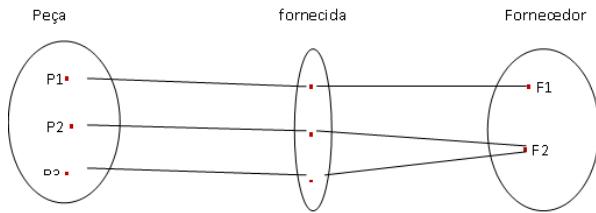
Os tipos de relacionamentos possuem certas restrições que limitam as combinações possíveis de entidades participando no relacionamento. Pode-se distinguir: **Razão de Cardinalidade** e **Restrição de Participação**. A **cardinalidade** especifica a quantidade de instâncias de relacionamento que uma entidade pode participar. As mais comuns são:

- **1:1** uma entidade *a* do tipo de entidade A está relacionada com, no máximo, uma entidade *b* do tipo de entidade B.
- **1:N** uma entidade *a* do tipo de entidade A pode estar relacionada com muitas entidades *b* do tipo de entidade B.

- **N:N** uma entidade *a* do tipo de entidade A pode estar relacionada com muitas entidades *b* do tipo de entidade B, e vice-versa.

A restrição de **participação** especifica se a existência de uma entidade depende do relacionamento com outra entidade. Podem ser:

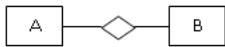
- **TOTAL**: todas as entidades do tipo de entidade A estão relacionadas com pelo menos uma entidade do tipo de entidade B.
- **PARCIAL**: se existem entidades do tipo de entidade A não relacionadas com uma entidade do tipo de entidade B.



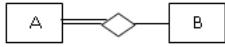
Todas as entidades do tipo A estão relacionadas com uma entidade do tipo B?

Se a resposta for **sim**, então a participação é **total**. Se a resposta for **não**, então a participação é **parcial**.

Indicamos a restrição de participação parcial com um traço de ligação simples entre a entidade que participa parcialmente do relacionamento. Veja:



A restrição de participação total é indicada com um traço duplo na ligação entre a entidade e o relacionamento. Veja a representação de participação total para este caso:



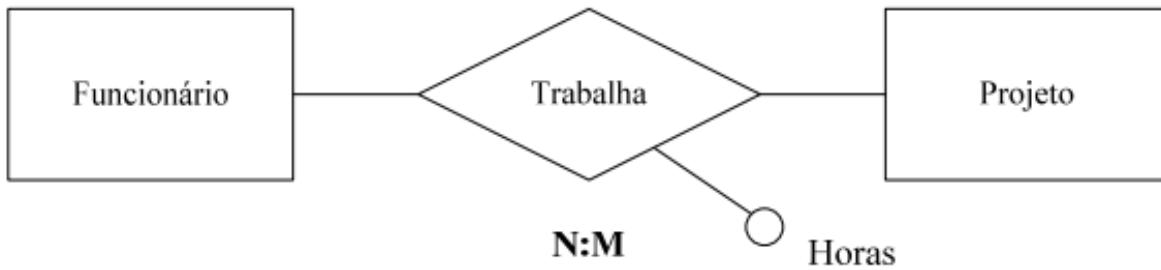
Atributos do relacionamento

Os tipos de relacionamento podem ter atributos. Se o valor do atributo é determinado pela combinação das entidades participantes então o atributo é do relacionamento.

Quando um determinado relacionamento possui atributos é também conhecido como relacionamento valorado.

Exemplo: “Pedro” trabalha no projeto “Alfa” 30 horas

- “Pedro”: Elemento do conjunto de valores do atributo Nome
- “Alfa”: Elemento do conjunto de valores do atributo Nome do Projeto da entidade Projeto.
- “Trabalha”: Ligação existente entre um funcionário e um projeto. Neste caso, este funcionário trabalha 30 horas neste projeto, porém este mesmo funcionário poderá trabalhar outro número de horas em outro projeto, assim como outro funcionário trabalha outro número de horas no mesmo projeto Alfa. Podemos concluir que 30 horas é o atributo que pertence ao Pedro no projeto Alfa

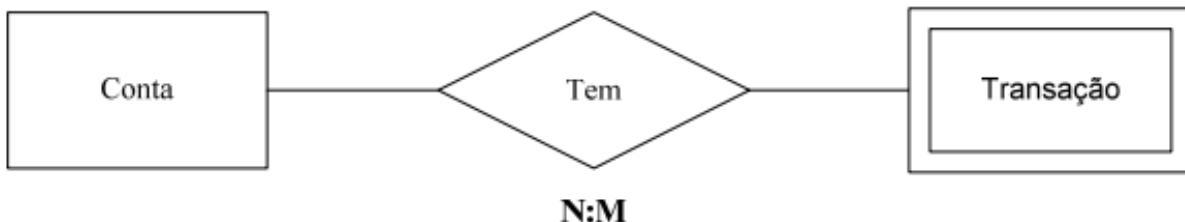


3.2.4 Tipo de entidade fraca

São tipos de entidades que podem não ter atributos chave e estão relacionadas a um tipo de entidade denominado proprietário da identificação. Um tipo de entidade fraca tem restrição de participação total com o proprietário.

Especificamente, se a existência da entidade X depende da existência da entidade Y, então diz-se que X é existencialmente dependente de Y, ou seja, X é uma entidade fraca. Operacionalmente, isto significa que se Y for removido, então X também será. A entidade Y é chamada de entidade dominante (ou proprietária) e X é chamada de entidade subordinada (ou fraca). Exemplo: Conta e Transação (Conta é proprietária e Transação é fraca)

Uma entidade fraca é representada por um duplo retângulo.



3.3 Diagramas Entidade-Relacionamento (DER), Convenções de Nomenclatura e Decisões de Projeto

As figuras (balões) ilustram exemplos da participação de tipos entidade em tipos relacionamento exibindo suas extensões - instâncias, em particular, de entidades, e instâncias de relacionamentos nos conjuntos de entidades e nos conjuntos de relacionamentos. Em diagramas ER, a ênfase está na representação do esquema em vez das instâncias. Isso é útil em projetos de banco de dados porque um esquema de banco de dados raramente é alterado, enquanto os conteúdos dos conjuntos de entidades frequentemente se alteram. Além disso, o esquema é normalmente mais fácil de exibir que toda a extensão de um banco de dados, pois é muito menor.

3.4 Notações para DER

Veja na figura a seguir a lista dos símbolos e principais notações utilizadas na construção de diagramas Entidade-Relacionamento, segundo Elmasri e Navathe(2011).

Símbolo	Significado
	Entidade
	Entidade fraca
	Relacionamento
	Relacionamento fraco
	Atributo
	Atributo chave
	Atributo multivalorado
	Atributo composto
	Atributo derivado
	Participação total de E ₂
	Razão de cardinalidade 1:N para E ₁ E ₂ em R
	Restrição estrutural (min, máx) da participação de E em R

Figura 3.2: Símbolos utilizados no DER e seus significados.

3.5 Exercícios

I Para cada cenário abaixo, faça o seguinte:

- 1 Liste as entidades que você identificar em cada um das situações.
- 2 Identifique e liste os atributos de cada entidade encontrada. Classifique cada atributo em: simples ou composto, monovalorado ou multivalorado, primitivo ou derivado, atributo nulo, atributo chave.
- 3 Liste os relacionamentos entre as entidades.

4 Crie o modelo entidade-relacionamento para o cenário.

- A Um berçário deseja informatizar suas operações. Quando um bebê nasce, algumas informações são armazenadas sobre ele, tais como: nome, data do nascimento, peso do nascimento, altura, a mãe deste bebê e o médico que fez seu parto. Para as mães, o berçário também deseja manter um controle, guardando informações como: nome, endereço, telefone e data de nascimento. Para os médicos, é importante saber: CRM, nome, telefone celular e especialidade.
- B 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 a compra, a data da compra, o valor total e os produtos comprados.
- C Uma Escola tem várias turmas. Uma turma tem vários professores, sendo que um professor pode ministrar aulas em mais de uma turma. Uma turma tem sempre aulas na mesma sala, mas uma sala pode estar associada a várias turmas (com horários diferentes).
- D Uma biblioteca deseja manter informações sobre seus livros. Inicialmente, quer armazenar para os livros as seguintes características: ISBN, título, ano editora e autores deste livro. Para os autores, deseja manter: nome e nacionalidade. Cabe salientar que um autor pode ter vários livros, assim como um livro pode ser escrito por vários autores. Cada livro da biblioteca pertence a uma categoria. A biblioteca deseja manter um cadastro de todas as categorias existentes, com informações como: código da categoria e descrição. Uma categoria pode ter vários livros associados a ela.
- E Uma firma vende produtos de limpeza, e deseja melhor controlar os produtos que vende, seus clientes e os pedidos. Cada produto é caracterizado por um código, nome do produto, categoria (ex. detergente, sabão em pó, sabonete, etc), e seu preço. A categoria é uma classificação criada pela própria firma. A firma possui informações sobre todos seus clientes. Cada cliente é identificado por um código, nome, endereço, telefone, status ("bom", "médio", "ruim"), e o seu limite de crédito. Guarda-se igualmente a informação dos pedidos feitos pelos clientes. Cada pedido possui um número e guarda-se a data de elaboração do pedido. Cada pedido pode envolver de um a vários produtos, e para cada produto, indica-se a quantidade pedida.

II Desenvolva o diagrama entidade-relacionamento para cada uma das situações abaixo. Defina as cardinalidades.

- 1 Um aluno realiza vários trabalhos. Um trabalho é realizado por um ou mais alunos.
- 2 Um diretor dirige no máximo um departamento. Um departamento tem no máximo um diretor
- 3 Um autor escreve vários livros. Um livro pode ser escrito por vários autores.
- 4 Uma equipe é composta por vários jogadores. Um jogador joga apenas em uma equipe.
- 5 Um cliente realiza várias encomendas. Uma encomenda diz respeito a um cliente.

3.6 Denominação dos construtores de Esquema

Ao projetar um esquema de banco de dados, a escolha dos nomes para tipos entidade, atributos, relacionamentos, tipos e (particularmente) papéis nem sempre é direta. Deveriam ser escolhidos

nomes que carregassem, tanto quanto possível, os significados dos diferentes construtores do esquema. Optamos por usar nomes no singular para os tipos entidade em vez do plural, porque o nome do tipo entidade se aplica a cada entidade em particular pertencente àquele tipo. Em nossos diagramas ER usaremos como convenção letras maiúsculas para tipos entidade e tipos relacionamento, a primeira letra maiúscula para os nomes dos atributos e letras minúsculas para os nomes dos papéis.

Como prática geral, dada uma descrição narrativa dos requisitos do banco de dados, os substantivos do texto tendem a originar nomes de tipos entidade e os verbos tendem a indicar nomes de tipos relacionamento. Os nomes dos atributos geralmente surgem de substantivos adicionais que descrevem os substantivos correspondentes aos tipos entidade.

Outra observação sobre a nomenclatura envolve a escolha dos nomes dos relacionamentos binários do esquema ER de modo a torná-los legíveis para a leitura da direita para a esquerda e de cima para baixo.

3.7 Decisões de projetos para o Projeto conceitual ER

Às vezes, é difícil decidir se um determinado conceito no minimundo deveria ser modelado como um tipo entidade, um atributo ou um tipo relacionamento. Nesta seção, daremos um breve resumo das diretrizes sobre qual construtor deveria ser escolhido em situações particulares.

Em geral, o processo de projeto do esquema deveria ser considerado um processo iterativo de refinamento, no qual um projeto inicial é criado e então iterativamente refinado até que o projeto mais satisfatório seja alcançado. Alguns dos refinamentos mais frequentemente usados são:

- Um conceito pode ser modelado primeiro como um atributo, e então pode ser refinado em um relacionamento porque foi determinado que o atributo é uma referência a outro tipo entidade. É comum que um par de tais atributos, que são o inverso um do outro, sejam refinados em um relacionamento binário.
- Analogamente, um atributo existente em vários tipos entidade pode ser promovido ou elevado a um tipo entidade independente. Por exemplo, suponha que vários tipos entidade de um banco de dados UNIVERSIDADE, como ALUNO, INSTRUTOR e CURSO, tenham, cada um, um atributo Departamento no projeto inicial; o projetista poderia criar, então, um tipo entidade DEPARTAMENTO com um único atributo NomeDept e relacioná-lo aos três tipos entidade (ALUNO, INSTRUTOR e CURSO) por meio de relacionamentos apropriados. Outros atributos/relacionamentos de DEPARTAMENTO poderão aparecer depois.
- Um refinamento inverso para o caso anterior poderá ser aplicado - por exemplo, se um tipo entidade DEPARTAMENTO existir no projeto inicial com um único atributo NomeDept, e estiver relacionado a somente um tipo entidade, ALUNO. Nesse caso, DEPARTAMENTO pode ser reduzido ou rebaixado a um atributo de ALUNO.

Mais adiante, discutiremos outros refinamentos relativos à especialização/generalização e a relacionamentos de graus mais altos. Além de refinamentos adicionais top-down (cima-para-baixo) e bottom-up (baixo-para-cima), que são comuns em grandes projetos de esquemas conceituais.

3.8 Exercícios

1 Elaborar um diagrama E-R para uma seguradora de automóveis

Entidades: Cliente, Apólice, Carro e Acidentes.

Requisitos:

- Um cliente pode ter várias apólices (no mínimo uma);
- Cada apólice somente dá cobertura a um carro;
- Um carro pode ter zero ou n registros de acidentes a ele.

Atributos:

- Cliente: Número, Nome e Endereço;
- Apólice: Número e Valor;
- Carro: Registro e Marca;
- Acidente: Data, Hora e Local;

2 Elaborar um diagrama E-R de um consultório clínico

Entidades: Médico, Paciente e Exame.

Requisitos: O banco de dados deverá armazenar informações sobre os vários exames de um determinado paciente, com o resultado e o valor pago (pode-se dar desconto para determinados pacientes);

Atributos:

- Médico: Número, Nome e Especialidade;
- Paciente: Número, Nome, Endereço;
- Tipo Exame, Aceita Convênio, Requisitos, Valor exame.

3 Elaborar um diagrama para uma Indústria.

Entidades: Peças, Depósitos, Fornecedor, Projeto, Funcionário e Departamento.

Requisitos:

- Cada Funcionário pode estar alocado a somente um Departamento;
- Cada Funcionário pode pertencer a mais de um Projeto;
- Um projeto pode utilizar-se de vários Fornecedores e de várias Peças;
- Uma Peça pode ser fornecida por vários Fornecedores e atender a vários Projetos;
- Um Fornecedor pode atender a vários Projetos e fornecer várias Peças;
- Um Depósito pode conter várias Peças;
- Deseja-se ter um controle do material utilizado por cada Projeto, identificando inclusive o seu Fornecedor. Gravar as informações de data de Início e Horas Trabalhadas no Projeto.

Atributos:

- Peças: Número, Peso e Cor;
- Depósito: Número e Endereço;
- Fornecedor: Número e Endereço;
- Projeto: Número e Orçamento;
- Funcionário: Número, Salário e Telefone;
- Departamento: Número e Setor.

4 Projetar um Banco de Dados satisfazendo as seguintes restrições e requisitos:

- Para um Vendedor, armazenar seu código, nome, endereço e comissão;
- Para um cliente, armazenar o seu código, nome, endereço, faturamento acumulado e limite de crédito. Além disso, armazenar o código e o nome do vendedor que o atende. Um vendedor pode atender muitos clientes, porém um cliente deve ter exatamente um vendedor;
- Para uma peça, armazenar seu código, descrição, preço quantidade em estoque e o número do armazém onde a peça está estocada. Uma peça somente pode estar estocada num único armazém. Para um armazém, armazenar seu código e endereço;
- Para um pedido, armazenar seu número, data, código, nome e endereço do cliente, que fez o pedido e o código do vendedor para cálculo da comissão. Além disso, para cada item do pedido armazenar o código da peça, quantidade e preço cotado. Há somente um cliente por pedido e um vendedor;
- O preço cotado no pedido pode ser mesmo que o preço corrente no arquivo de peças, mas não necessariamente.

5 Projete um diagrama ER para acompanhar as informações sobre a votação realizada pelos representantes do congresso nacional durante as seções dos últimos dois anos. O banco de dados precisa acompanhar o nome de cada estado brasileiro e a região do estado. Cada membro do congresso no Congresso Nacional é descrito através de seu nome e inclui o estado representado, data de início, quando tomou posse e o partido político ao qual pertence. O banco de dados acompanha cada projeto de lei (isto é, lei proposta) e inclui o nome do projeto de lei, a data da votação do projeto e se o projeto foi aprovado ou reprovado e o responsável (o(s) congressista(s) que apoio ou apoiaram ? isto é, propôs ou propuseram ? o projeto de lei). O banco de dados acompanha como cada congressista votou em cada projeto (o domínio do atributo voto é SIM, NÃO, ABSTENÇÃO ou AUSENTE). Projete um diagrama do esquema ER para a aplicação acima. Declare de forma clara qualquer pressuposto que você estabeleça.

6 Um banco de dados está sendo construído para acompanhar os times e as partidas de uma liga de esportes. Um time possui um número de jogadores, dos quais nem todos participam em cada partida. Deseja-se acompanhar os jogadores que participam em cada partida para cada time, as posições em que jogaram naquela partida e o resultado da partida. Tente projetar um diagrama do esquema ER para esta aplicação, declarando quaisquer pressupostos que você estabeleça. Escolha seu esporte favorito.

7 Uma pequena locadora de vídeo possui em torno de 2.000 fitas de vídeo, cujo empréstimo deve ser controlado. Cada fita possui um número. Para cada filme, é necessário saber seu título e sua categoria (comédia, drama, aventura, terror, clássico e musical). Cada filme recebe um identificador próprio. Para cada fita é controlado que filme ela contém. Para cada filme há pelo menos uma fita, e cada fita contém somente um filme. Alguns poucos filmes necessitam de duas fitas.

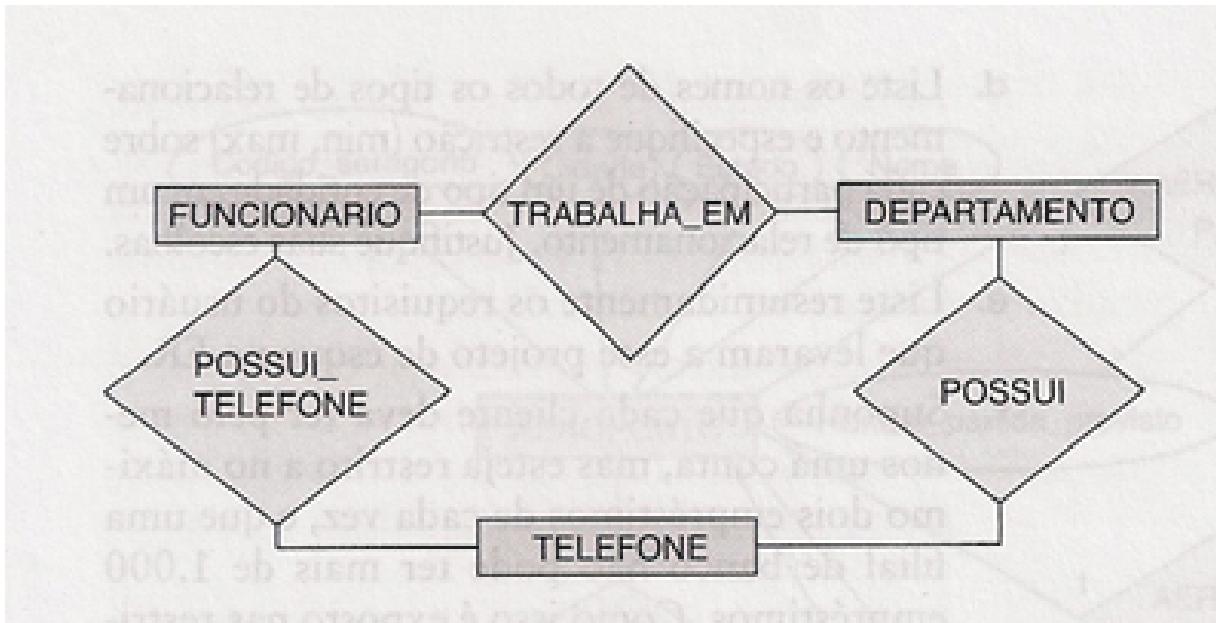
Os clientes podem desejar encontrar os filmes estrelados pelo seu ator predileto. Por isso, é necessário manter a informação dos atores que estrelam em cada filme. Nem todo filme possui estrelas. Para cada ator os clientes às vezes desejam saber o nome real, bem como a data de nascimento. A locadora possui muitos clientes cadastrados. Somente clientes cadastrados podem alugar fitas. Para cada cliente são necessários saber seu prenome e seu sobrenome, seu telefone e seu endereço. Além disso, cada cliente recebe um número de associado. Finalmente, desejamos

saber as fitas emprestadas para cada cliente. Um cliente pode ter várias fitas em um instante de tempo. Não são mantidos registros históricos de aluguéis.

- 8 Considere o seguinte conjunto de requisitos para um banco de dados UNIVERSIDADE, que é usado para registrar os históricos dos alunos.

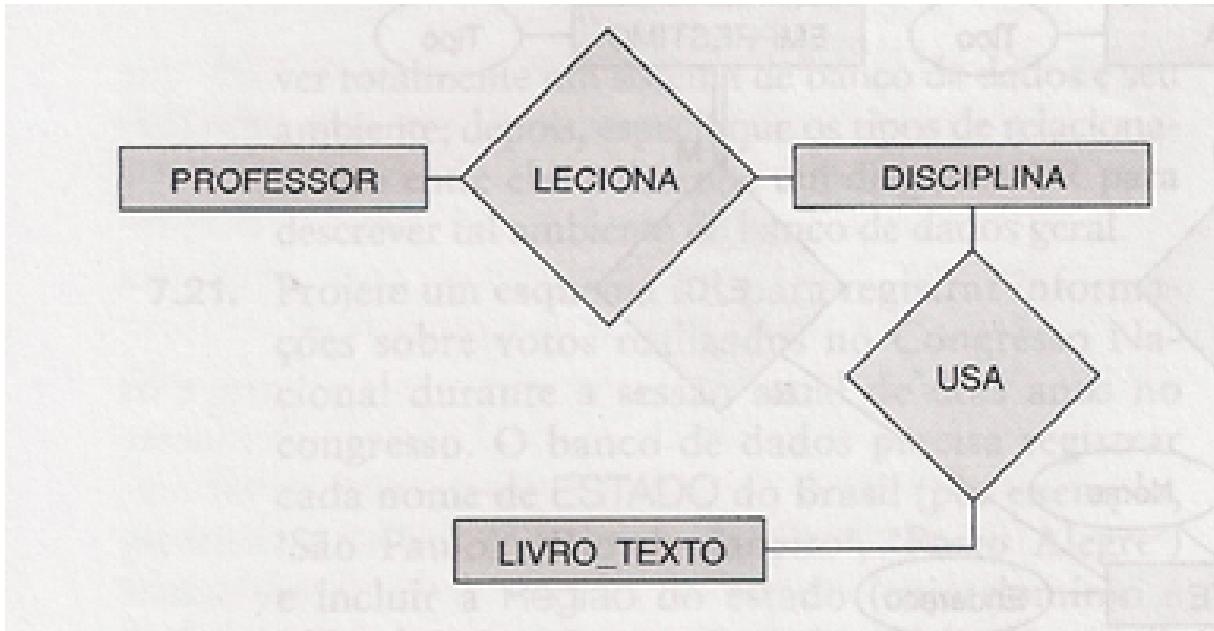
A universidade registrar o nome, número de aluno, número do CPF, endereço atual e com seu número de telefone fixo, endereço permanente e com seu número de telefone fixo, data de nascimento, sexo, nível no curso (novato, segundo ano, ..., formado), departamento principal, departamento secundário (se houver) e programa de formação (graduação, mestrado, ..., doutorado) de cada aluno. Algumas aplicações do usuário precisam se referir à cidade, estado e CEP do endereço permanente do aluno e ao sobrenome do aluno. O número do CPF e o número de aluno possuem valores exclusivos para cada um deles. Cada departamento é descrito por um nome, código de departamento, número de escritório e número de telefone comercial. Nome e código possuem valores exclusivos para cada departamento. Cada disciplina tem um nome, descrição, número de disciplina, número de horas por semestre, nível e departamento que oferece. O valor do número da disciplina é exclusivo para cada uma delas. Cada turma tem um professor, semestre, ano, disciplina e número de turma. O número de turma distingue as turmas da mesma disciplina que são lecionadas durante o mesmo semestre/ano; seus valores são 1, 2, 3, ..., até o número de turmas selecionadas durante cada semestre. Um relatório de notas tem um aluno, turma, nota com letra (conceito) e nota numérica (0 a 10).

- 9 Considere o diagrama ER da figura abaixo. Suponha que um funcionário possa trabalhar em até dois departamentos ou não possa ser atribuído a qualquer departamento. Suponha que cada departamento deva ter um e possa ter até três números de telefone. Forneça restrições (min, máx) sobre este diagrama. *Indique claramente quaisquer suposições adicionais que estiver fazendo.*



- 10 Considere o diagrama da figura abaixo. Suponha que uma disciplina possa ou não usar um livro-texto, mas que um texto por definição é um livro que é usado em alguma disciplina. Uma disciplina não pode usar mais de cinco livros. Os professores lecionam de duas a quatro disciplinas. Forneça restrições (min, máx) sobre esse diagrama. *Indique claramente quaisquer suposições adicionais*

que estiver fazendo. Se acrescentarmos o relacionamento ADOTADO, para indicar o(s) livro(s)-texto que um professor utiliza para uma disciplina, ele deverá ser um relacionamento binário entre PROFESSOR e LIVRO_TEXTO, ou um relacionamento ternário entre todos os três tipos de entidade? Que restrições (min, máx) você incluiria? Por quê?



Lembrete: Num relacionamento ternário, as 3 entidades estão associadas simultaneamente, sendo que a cardinalidade, neste caso, refere-se à quantidade de ocorrências de uma entidade em relação ao par das outras entidades.

11 Faça um diagrama ER para um sistema de controle bancário a partir das seguintes informações:

- Cada banco tem um nome e um código de identificação no sistema bancário. Ambos são únicos no sistema;
- Cada banco possui um conjunto de agências, onde os clientes abrem suas contas. Cada agência possui um código de identificação, que é único dentro de cada banco (agências de bancos diferentes podem ter o mesmo código), um nome e um endereço (logradouro, nº, complemento, bairro, cidade, cep, estado);
- Dos clientes são necessárias as seguintes informações: cpf, nome e endereço (logradouro, nº, complemento, bairro, cidade, cep, estado). Considere que o cpf identifica unicamente cada cliente;
- As contas são somente do tipo conta-corrente e possuem um número de identificação, que é único dentro do sistema, e um saldo. As contas podem ser individuais (de um único cliente) ou conjuntas (pertencerem a mais de um cliente). Um cliente pode ter mais de uma conta em uma mesma agência. É necessário saber também a data em que cada conta foi aberta. As contas possuem movimentações (ou transações) para as quais deve-se manter um controle histórico. Cada movimentação possui um número de identificação, que é único dentro do sistema, a data e o horário em que ela ocorreu, o tipo de movimentação (débito ou crédito), a descrição da movimentação e o valor movimentado.

12 Faça um diagrama ER para um sistema de controle de estoque, compra e venda de uma loja de produtos de informática, a partir das seguintes informações:

- A loja possui um cadastro de produtos. Cada produto possui um código (único no sistema), uma descrição e um preço de venda. Também é necessário saber-se a quantidade disponível em estoque de cada produto;
- A loja vende produtos para seus clientes através de pedidos, que do ponto de vista da loja são os pedidos de venda. Em cada pedido de venda consta um número de identificação do pedido, a data do pedido, os dados do cliente (cpf, nome, endereço (logradouro, nº, complemento, bairro, cidade, cep, estado) e telefones de contato) e a relação de produtos pedidos, incluindo o preço de venda e a quantidade pedida. Cada pedido de venda também possui uma situação indicando se ele está pendente, já foi entregue ou foi cancelado.
- A loja paga mensalmente os vendedores por meio de comissão de venda. Cada vendedor possui um número de registro na loja, um nome e uma data de admissão. Cada vendedor recebe um percentual sobre as vendas que ele efetua. Esse percentual varia de vendedor para vendedor dependendo do tempo de trabalho na loja.
- Para comprar um produto, a loja faz orçamentos com diversos fornecedores. Em cada orçamento consta os dados do fornecedor (cnpj, razão social, nome fantasia, endereço (logradouro, nº, complemento, bairro, cidade, cep, estado) e telefones de contato), os dados do produto cotado, o preço de compra, a data de cotação, a data de validade do preço cotado e o prazo de entrega. Com base nos orçamentos, são feitos pedidos de compra para os fornecedores que oferecem as melhores condições. Em cada pedido de compra consta um número de identificação do pedido, a data do pedido, os dados do fornecedor e a relação de produtos pedidos, incluindo o preço e a quantidade pedida. Cada pedido possui também uma situação indicando se ele está pendente, já foi entregue ou foi cancelado. Não é necessário relacionar os pedidos de compra aos orçamentos;

4 O Modelo Entidade-Relacionamento Estendido - EER

Os conceitos de modelagem ER que discutimos até aqui são suficientes para representar muitos esquemas de banco de dados para aplicações tradicionais, que incluem diversas aplicações de processamento de dados no comércio e indústria.

No entanto, desde o final da década de 1970, os projetistas têm tentado projetar esquemas de BD mais precisos e que refletem as propriedades de dados e restrições com mais fidelidade. Aplicações mais novas da tecnologia de BD, como para projetos de engenharia e manufatura (CAD/CAM), telecomunicações, sistemas complexos, sistemas de informações geográficas (GIS), entre outras.

Isso levou ao desenvolvimento de conceitos adicionais de *modelagem semântica de dados*, que foram incorporados em *modelos de dados conceituais*, como o modelo ER.

Analisaremos os recursos que foram propostos para modelos de dados semânticos e mostraremos como o modelo ER pode ser melhorado para incluir esses conceitos, levando ao modelo de dados Entidade-Relacionamento Estendido (EER).

4.1 Subclasse, Superclasse e Herança

O modelo Entidade-Relacionamento Estendido (EER), inclui todos os conceitos de modelagem do modelo ER. Além disso, inclui os conceitos de **subclasse** e **superclasse** e os conceitos relacionados de **especialização** e **generalização**. Outro conceito incluído no modelo EER é o de uma **categoria** ou **tipo de união** usado para representar uma coleção de objetos (entidades), que é a *união* de objetos de diferentes tipos de entidade. Associados a esses conceitos está o importante mecanismo de **herança de atributo** e **relacionamento**.

Em muitos casos, um tipo de entidade tem vários subconjuntos de entidades que são significativos para a aplicação.

- Exemplo: as entidades de um tipo de entidade **Funcionário** podem ser agrupadas em **Secretária**, **Engenheiro**, **Gerente**, **Funcionário_assalariado**, **Funcionário_horista**, etc.

Cada um dos subconjuntos é chamado de **subclasse** do tipo de entidade Funcionário, e o tipo de entidade Funcionário é chamado de **superclasse**.

Uma entidade não pode existir no banco de dados como membro somente de uma subclasse, ela deve também ser membro da superclasse. Não é necessário que toda entidade em uma superclasse seja membro de alguma subclasse.

Como uma entidade na subclasse representa a mesma entidade no mundo real da superclasse, então ela possui valores de seus atributos específicos bem como valores de seus atributos como um membro da superclasse.

- Uma entidade que pertence a uma subclasse herda todos os atributos da superclasse.

A entidade também **herda** todos os relacionamentos dos quais a superclasse participa. E pode ter seus próprios relacionamentos como subclasse.

4.2 Generalização e Especialização

Especialização é o processo de definir um conjunto de subclasses de um tipo de entidade (superclasse).

No exemplo da Figura 4.1 temos:

- O conjunto de subclasses Secretário, Técnico e Engenheiro é uma especialização de Funcionário que distingue as entidades baseado no tipo de trabalho.

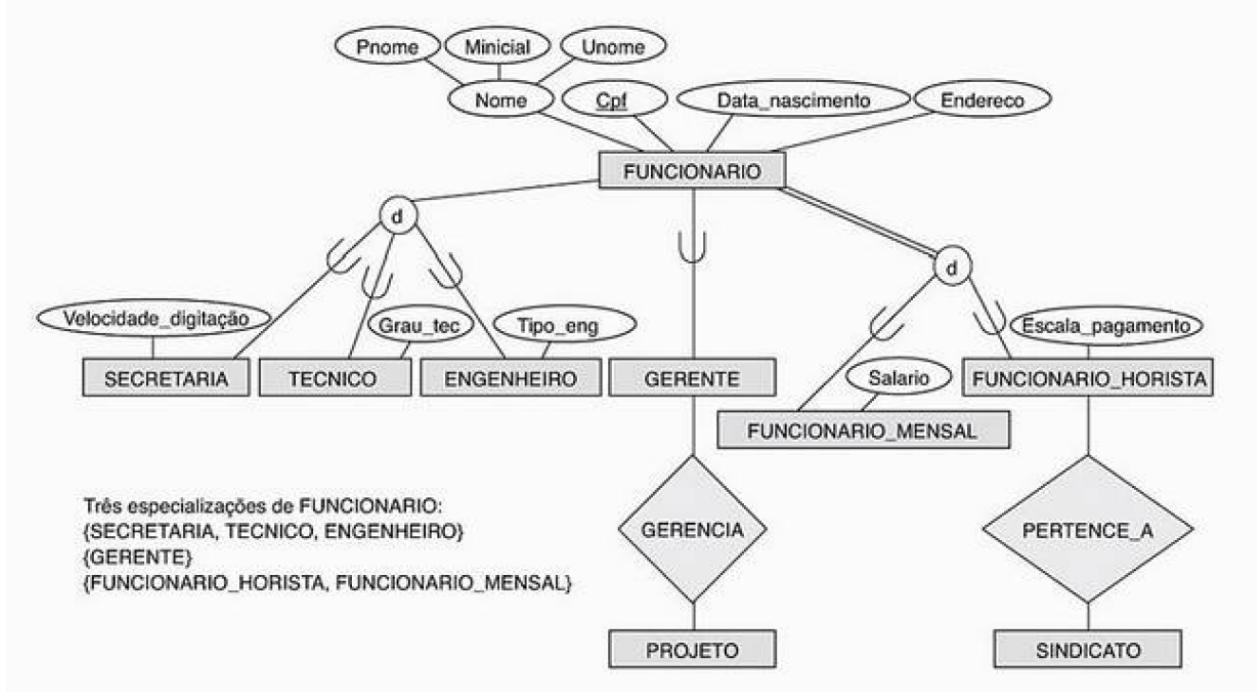


Figura 4.1: Notação EER para representar subclasses e especialização

- O conjunto de subclasses Mensalista e Horista é uma especialização de Funcionário que distingue as entidades baseado na forma de pagamento.
- Somente as entidades da classe Engenheiro possuem o atributo tipo_eng (tipo de engenheiro). Todas as entidades em Secretário, Técnico, Engenheiro, Gerente, Assalariado e Horista possuem os atributos cpf e nome.
- Somente as entidades da classe Gerente podem participar do tipo de relacionamento *gerencia*.

Generalização é o processo inverso da Especialização, ou seja, as diferenças entre dois ou mais tipos de entidades são suprimidas, suas características comuns são identificadas, e é feita uma generalização em uma única superclasse da qual os tipos originais são subclasses especiais.

Na Figura 4.2 podemos ver {CARRO, CAMINHÃO} como uma especialização de VEÍCULO, em vez de VEÍCULO como uma generalização de CARRO e CAMINHÃO. De modo semelhante, na Figura 4.1, podemos ver FUNCIONÁRIO como uma generalização de SECRETÁRIA, TÉCNICO e ENGENHEIRO.

Existe uma notação diagramática para distinguir generalização de especialização, usada em algumas metodologias. Uma seta apostando para a superclasse generalizada representa uma generalização, ao passo que setas apostando para subclasses especializadas representam uma especialização.

4.3 Restrições sobre Especialização e Generalização

Em geral, podemos ter várias especializações definidas no mesmo tipo de entidade (ou superclasse) como mostra a Figura 4.1. Neste caso, as entidades podem pertencer a subclasses em cada uma das

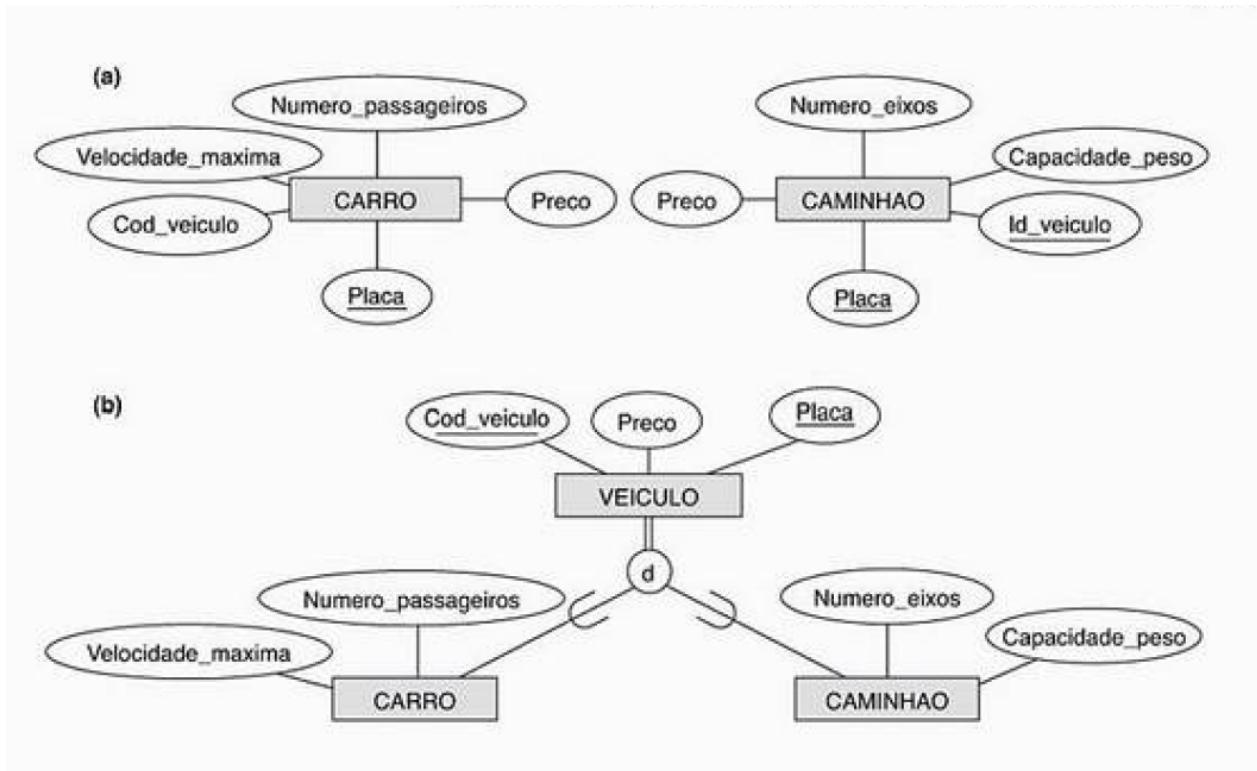


Figura 4.2: Exemplos de generalização

especializações. Contudo, uma especialização também pode consistir em uma única subclasse, como a especialização {GERENTE} na Figura 4.1; em tal caso, não usamos a notação de círculo.

Em algumas especializações, podemos determinar exatamente as entidades que se tornarão membros de cada subclasse ao colocar uma condição sobre o valor de algum atributo da superclasse. Essas subclasses são chamadas **subclasses definidas por predicado** (ou **definidas por condição**). Por exemplo, se o tipo de entidade FUNCIONÁRIO tiver um atributo Tipo_emprego, como mostra a Figura 4.3, podemos especificar a condição de membro na subclasse SECRETÁRIA pela condição (`Tipo_emprego = 'Secretária'`), que chamamos de **predicado de definição** da subclasse. Essa condição é uma restrição que especifica exatamente que aquelas entidades do tipo de entidade FUNCIONÁRIO, cujo valor de atributo para Tipo_emprego é 'Secretária', pertencem à subclasse. Apresentamos uma subclasse definida pelo predicado escrevendo a condição de predicado ao lado da linha que conecta a subclasse ao círculo de especialização.

Se *todas* as subclasses em uma especialização tiverem sua condição de membro no *mesmo* atributo da superclasse, a própria especialização é chamada de **especialização definida por atributo**, e o atributo é chamado de **atributo de definição** da especialização. Neste caso, todas as entidades com o mesmo valor para o atributo pertencem à mesma subclasse. Apresentamos uma especialização definida por atributo colocando o nome do atributo de definição ao lado do arco que vai do círculo à superclasse, como mostra a Figura 4.3.

Quando não temos uma condição para determinar os membros em uma subclasse, diz-se que esta é **definida pelo usuário**. A condição de membro nessa subclasse é determinada pelos usuários do banco de dados quando eles aplicam a operação para incluir uma entidade à subclasse; logo, a condição de membro é especificada individualmente para cada entidade pelo usuário, e não por

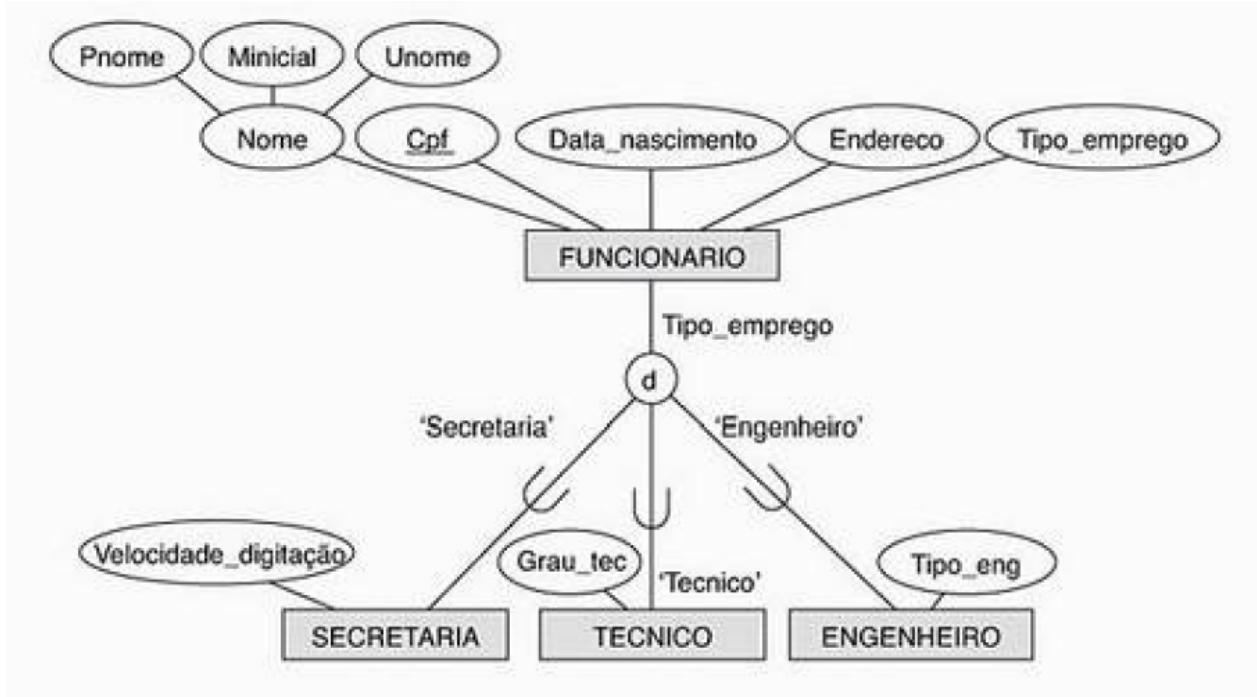


Figura 4.3: Notação do diagrama EER para uma especialização definida por atributo

qualquer condição que possa ser avaliada automaticamente.

Duas outras restrições podem se aplicar a uma especialização.

Restrição de disjunção (ou **desconexão**), que especifica que as subclasses da especialização devem ser disjuntas.

Disjunção: uma entidade pode ser membro de, no máximo, uma das subclasses da especialização.

Representada pela letra **d** dentro do círculo da disjunção. Exemplo: Especializações da Figura 4.1.

Sobreposição: uma entidade pode ser membro de mais de uma subclasse da especialização. Representada pela letra **o** (*overlapping*) dentro do círculo da disjunção. Exemplo: Especialização da Figura 4.4.

Restrição de completude (ou **totalidade**) que pode ser total ou parcial.

Total: toda entidade na superclasse deve ser membro de pelo menos uma subclasse na especialização. Representada pelas linhas duplas ligando a superclasse ao círculo. Exemplo: A especialização **{FUNCIONÁRIO_HORISTA, FUNCIONÁRIO_MENSAL}** da Figura 4.1 é uma especialização total de **FUNCIONÁRIO**.

Parcial: uma entidade na superclasse não precisa ser membro de nenhuma subclasse na especialização. Representada pela linha simples ligando a superclasse ao círculo. Exemplo: Se algumas entidades **FUNCIONÁRIO** não pertencerem a nenhuma das subclasses **{SECRETÁRIA, ENGENHEIRO, TÉCNICO}** nas Figuras 4.1 e 4.3.

Observe que as restrições de disjunção e completude são independentes.

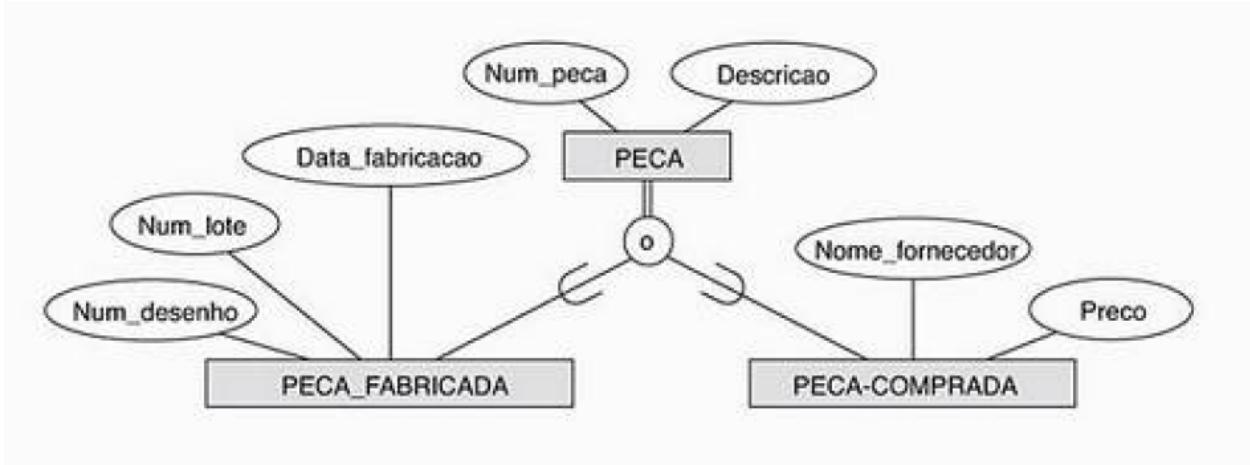
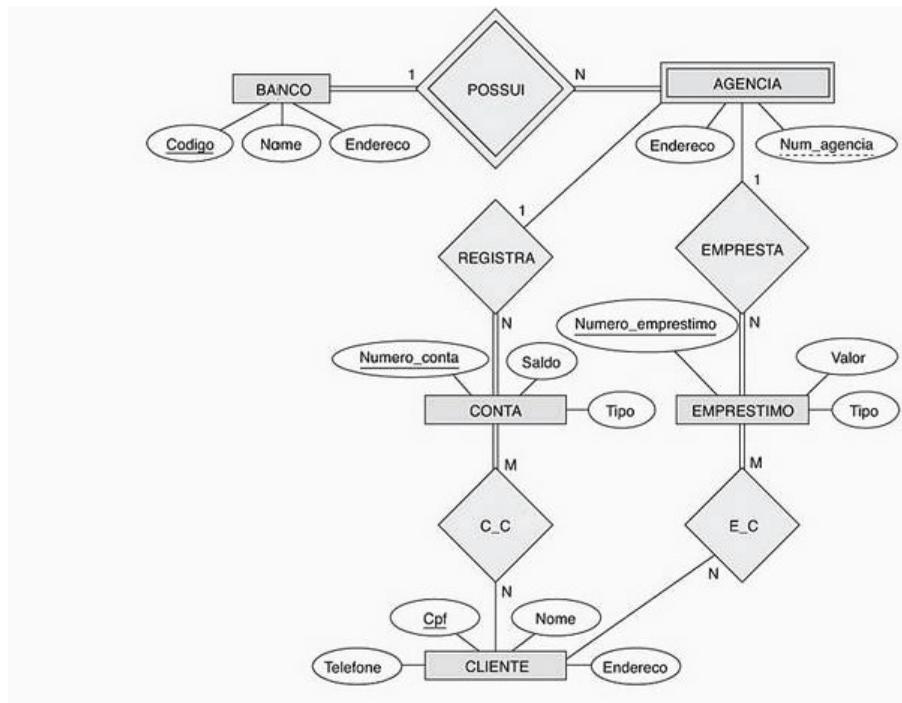


Figura 4.4: Exemplo de especialização sobreposta

4.4 Exercícios

1. Considere o seguinte diagrama ER e suponha que seja necessário registrar diferentes tipos de CONTAS (CONTA_POUPANCA, CONTA_CORRENTE, ...) e EMPRESTIMOS (EMPRESTIMO_CARRO, EMPRESTIMO_HABILITACAO, ...). Suponha que também se deseja registrar cada uma das TRANSACOES de CONTA (depósitos, saques, cheques, ...) e os PAGAMENTOS de EMPRESTIMO; ambos incluem o valor, data e hora. Modifique o esquema BANCO, usando os conceitos de ER e EER de especialização e generalização. Indique quaisquer suposições que você fizer sobre os requisitos adicionais.



2. Identifique todos os conceitos importantes apresentados no estudo de caso do banco de dados

de biblioteca descrito a seguir. Em particular, identifique as abstrações de classificação (tipos de entidade e tipos de relacionamento), especialização/generalização. Especifique as restrições(mín, max) sempre que possível. Liste detalhes que afetarão o eventual projeto, mas que não têm significado sobre o projeto conceitual. Desenhe um diagrama EER do banco de dados de biblioteca.

Estudo de caso: a Georgia Tech Library (GTL) tem aproximadamente 16 mil usuários, 100 mil títulos e 2,50 mil volumes (uma média de 2,5 cópias por livro). Cerca de 10% dos volumes estão emprestados a qualquer momento. Os bibliotecários garantem que os livros estejam disponíveis quando os usuários querem apanhá-los emprestado. Além disso, os bibliotecários precisam saber quantas cópias de cada livro existem na biblioteca ou emprestadas a qualquer momento. Um catálogo de livros está disponível on-line, listando livros por autor, título e assunto. Para cada título da biblioteca, uma descrição do livro é mantida no catálogo, que varia de parágrafos a várias páginas. Os bibliotecários de referência desejam poder acessar essa descrição quando os usuários solicitarem informações sobre um livro. O pessoal da biblioteca inclui o bibliotecário chefe, bibliotecários de referência, pessoal de despacho e assistentes de bibliotecário.

Os livros podem ser emprestados por 21 dias. Os usuários têm permissão para pegar apenas cinco livros de uma só vez. Os usuários normalmente retornam os livros dentro de três a quatro semanas. A maioria dos usuários sabe que têm uma semana de tolerância antes que uma nota seja enviada para eles, e por isso tentam retornar os livros antes que o período de tolerância termine. Cerca de 5% dos usuários precisa receber lembretes para devolver os livros. A maioria dos livros atrasados é retornada dentro de um mês da data de vencimento. Aproximadamente 5% dos livros atrasados são mantidos ou nunca são retornados. Os membros mais ativos da biblioteca são definidos como aqueles que pegam livros emprestados pelo menos dez vezes durante o ano. Um por cento dos usuários que mais utilizam empréstimos realizam 15% dos empréstimos, e os maiores 10% dos usuários realizam 40% dos empréstimos. Cerca de 20% dos usuários são totalmente inativos por nunca terem emprestado livros.

Para tornar-se um usuário da biblioteca, os candidatos preenchem um formulário incluindo seu CPF, endereço de correspondência do campus e da residência, e números de telefone. Os bibliotecários emitem um cartão numerado, legível à máquina, com foto do usuário. Esse cartão tem validade de quatro anos. Um mês antes de um cartão expirar, uma nota é enviada ao usuário para fazer renovação. Os professores do instituto são considerados usuários automáticos. Quando um novo usuário do corpo docente entra para o instituto, sua informação é puxada dos registros de funcionário e um cartão da biblioteca é remetido ao seu endereço no campus. Os professores têm permissão para retirar livros por intervalos de três meses, e possuem um período de tolerância de duas semanas. As notas de renovação para os professores são enviadas para seu endereço no campus.

A biblioteca não empresta alguns livros, como livros de referência, livros raros e mapas. Os bibliotecários precisam diferenciar livros que podem ser emprestados daqueles que não podem. Além disso, eles possuem uma lista de alguns livros em que estão interessados em adquirir, mas não conseguem obter, como livros raros ou que estão esgotados, e livros que foram perdidos ou destruídos, mas não substituídos. Os bibliotecários precisam ter um sistema que registre os livros que não podem ser emprestados bem como os livros que eles estão interessados em adquirir. Alguns livros podem ter o mesmo título; portanto, o título não pode ser usado como um meio de identificação. Cada livro é identificado por seu International Standard Book Number (ISBN), um código internacional exclusivo atribuído a todos os livros. Dois livros com o mesmo título podem ter diferentes ISBNs se estiverem em diferentes idiomas ou tiverem

diferentes encadernações (capa dura ou brochura). As edições de um mesmo livro possuem ISBNs diferentes.

3. Faça um diagrama ER/EER para um sistema de um museu de artes, a partir das seguintes informações:

- O museu tem uma coleção de obras de arte. Cada obra de arte tem um número de identificação, um artista (se conhecido), um ano (quando a obra foi criada, se conhecido), um título e uma descrição. As obras de arte são categorizadas como pintura, escultura, estátua e outros (obras que não são dos três tipos anteriores). Uma pintura tem um tipo (óleo, água etc), um material no qual foi desenhada (papel, madeira, canvas etc) e um estilo (moderna, abstrata etc). Uma escultura tem o material em que foi criada (madeira, pedra etc), a altura, o peso e o estilo. Uma obra de arte na categoria outro item um tipo (impressão, foto etc) e um estilo;
- As obras de arte também são categorizados como coleção permanente que são de propriedade do museu (neste caso guarda-se informações sobre data de aquisição, se a obra está em exposição ou está guardada e o custo da obra) ou como emprestados (que guarda informações sobre a coleção a que ela pertence, a data de empréstimo e a data de devolução);
- As obras de arte também guardam informações sobre o seu país/cultura de origem (italiana, egípcia etc) e a época em que foram criadas (renascimento, modernismo, etc);
- O museu mantém informações sobre os artistas (se conhecidos): nome, data de nascimento, data de falecimento (se não está vivo), país de origem, época, principal estilo e descrição. O nome é tido como único;
- Ocorrem diferentes exibições, cada uma tem nome, data de início, data de término e uma relação de todas as obras de arte que foram mostradas durante a exibição;
- Das coleções com que o museu interage também são mantidas informações sobre nome (único), tipo (museu, pessoal etc), descrição, endereço, telefone e pessoa de contato.

4. Projete um diagrama ER/EER para o banco de dados de um pequeno aeroporto particular, com base nos requisitos abaixo:

- Cada aeronave tem um número de registro, é de um tipo de avião em particular e é mantido em um hangar em particular.
- Cada tipo de avião tem um número de modelo, uma capacidade e um peso.
- Cada hangar tem um número, uma capacidade e um local.
- O banco de dados também registra os proprietários de cada avião e os funcionários que fazem a manutenção do avião.
- Cada avião passa por manutenção muitas vezes; logo, ela é relacionada a uma série de registros de serviço. Um registro de serviço inclui como atributos a data da manutenção, o número de horas gastas no trabalho e o tipo de trabalho realizado.
- Um proprietário é uma pessoa ou uma corporação.
- Tanto pilotos quanto funcionários são subclasses de pessoa. Cada piloto tem atributos específicos de número de licença e restrições; cada funcionário tem atributos específicos de salário e turno trabalhado.

- Todas as pessoas no banco de dados possuem dados mantidos sobre seu número de CPF, nome, endereço e número de telefone.
- Para uma corporação, os dados mantidos incluem nome, endereço e número de telefone.
- O banco de dados também registra os tipos de aviões que cada piloto é autorizado a voar e os tipos de aviões em que cada funcionário pode realizar o trabalho de manutenção.

4.5 Estudos de caso

4.5.1 Estudo de caso 1: Sistema de Reservas em Hotel

Um hotel local precisa de um sistema que mantenha os registros de suas reservas (futuras, atuais e arquivadas), quartos e hóspedes. Um quarto pode ser de um tipo e uma faixa de preço particular. Os preços dos quartos variam de quarto pra quarto (dependendo de seu tipo, facilidades disponíveis, faixas, etc.) e de temporada para temporada (dependendo do período do ano). Uma reserva de quarto pode incluir um quarto e mais que um cliente.

Diversas questões podem ser feitas a partir dos dados do banco de dados, tais como:

- Quantos quartos estão atualmente disponíveis para reserva?
- Quais facilidades estão disponíveis em quartos particulares?
- Quais hóspedes estão reservados para uma semana?

4.5.2 Estudo de caso 2: Sistema de controle de orçamento

Na modelagem dos dados para o sistema de orçamentos iremos de considerar o seguinte cenário:

Cenário: O Sr. X, desenvolve mensalmente uma planilha com a previsão de suas contas de despesa e receita no intuito de melhor organizar sua gestão orçamentária. Para melhor compreender a necessidade do Sr. X, analisemos a Figura 4.5.2.

A	B	C	D	E
1 Mês/Ano		set/17		out/17
	Previsto	Realizado	Previsto	Realizado
3				
4 Receitas				
5 Salários	R\$1.000,00	R\$1.000,00	R\$1.000,00	
6 Outros Recebimentos	R\$500,00	=200+200	R\$500,00	
7 Total Receitas	R\$1.500,00	R\$1.400,00	R\$1.500,00	
8				
9 Despesas				
10 Despesas Fixas				
11 Aluguel	R\$200,00	R\$200,00	R\$200,00	
12 Telefone	R\$100,00	R\$100,00	R\$100,00	
13 Água	R\$20,00	R\$20,00	R\$20,00	
14 Total Despesas Fixas	R\$320,00	R\$320,00	R\$320,00	
15				
16 Despesas Variáveis				
17 Supermercado	R\$340,00	R\$340,00	R\$350,00	
18 Lazer	R\$200,00	R\$355,00	R\$200,00	
19 Total Despesas Variáveis	R\$540,00	R\$695,00	R\$550,00	
20				
21 Saldo (Receita - Despesa)	R\$640,00	R\$385,00	R\$630,00	
22				

Figura 4.5: Planilha de orçamento mensal

Na Figura 4.5.2, constatamos as seguintes características presentes na planilha:

- 1 Ela possui contas de Receitas e Despesas, que são subdivididas em despesas fixas e variáveis. As Receitas consistem em todos os valores que podem ser recebidos pelo Sr. X, as Despesas Fixas são aquelas que irão acontecer todo mês e com valor pré-determinado e as Despesas Variáveis não têm um valor fixo e os valores podem oscilar de um mês para outro;
- 2 Apresenta duas colunas todo mês, uma de valor Previsto e outra de Realizado. O valor previsto consiste no orçamento atribuído ou previsão de valor para a conta, já o Realizado consiste no valor que efetivamente foi lançado (gasto ou recebido) em determinada conta;
- 3 Os valores previstos e realizados são separados por mês. O valor previsto é lançado uma única vez no mês para cada conta orçada. Já o valor realizado pode receber mais de um lançamento (ver coluna de receita C6 ? Outros recebimentos do mês set/2008 na Figura 5);
- 4 A linha Saldo nada mais é do que um cálculo do que foi recebido (Receita) subtraindo o que foi pago (Despesa).
- 5 Cada grupo de contas de Despesa e Receita receberá um subtotal.

Apesar de não estar explícito na planilha, é importante considerar ainda o que segue:

- 6 Em todo mês poderão surgir novas contas, devendo portanto prever a possibilidade de inserção destas novas contas;
- 7 Eventualmente contas não orçadas irão acontecer, ou seja, poderão receber lançamentos mesmo sem previsão.
- 8 Contas orçadas podem também não receber lançamentos em determinado mês;
- 9 Mesmo que as contas de Receita não apresentem divisões, não significa que estas não os terão futuramente.

4.5.3 Estudo de caso 3: Sistema de Zoo

Cenário considerado

Um zoológico planeja adquirir um sistema para gerenciar as suas tarefas diárias. O sistema para esse zoológico deve ser capaz de cadastrar os seus animais, bem como os seus funcionários.

O sistema deve reconhecer também a que classe um determinado animal pertence: mamífero, réptil ou ave. Além disso, cada classe deve conter uma descrição técnica sobre as suas características específicas.

Os animais do zoológico devem ser identificados por um código de identificação, devendo o sistema registrar o seu nome, espécie, cor e altura. Cada animal é mantido em algum tipo de container e o sistema deve saber o seu tipo, por exemplo: um poço, uma jaula, um viveiro, um tanque, etc. Cada container fica localizado em uma ala do zoológico e o sistema deve indicar qual é essa ala onde ele se encontra para facilitar o agendamento de atividades.

Os funcionários que trabalham nesse zoológico podem ser veterinários, cuidadores de animais, zeladores ou trabalharem em setores administrativos. Todos os funcionários devem conter nome, data de nascimento, CPF, RG, endereço completo e o cargo que desempenham. No caso dos veterinários, deve conter ainda o CRMV (Carteira do Conselho Regional de Medicina Veterinária).

O sistema deve possuir um módulo para agendar dia e hora para as consultas de cada animal com os veterinários, tendo em vista que cada animal deve ser tratado sempre pelo mesmo veterinário, para que ele possa fazer acompanhamentos em longo prazo com o seu paciente, ressaltando que um veterinário pode atender, no máximo, 15 animais, sendo este o número ideal para que o veterinário consiga manter a qualidade do seu trabalho.

O sistema deve também ser capaz de agendar dia e horário específico para a limpeza dos *containers* dos animais.

4.5.4 Estudo de caso 4: Sistema de Saúde

Para este projeto foi escolhido como cenário um minimundo dentro de um projeto maior, a criação do Sistema Nacional de Saúde da Pessoa. O escopo definido para este projeto trata apenas a interação entre o paciente e o médico a nível ambulatorial.

No nível ambulatorial, as interações entre o médico e o paciente são estabelecidas através de consultas que são registradas em atendimentos. Para o cadastro de pacientes são indispensáveis o código do SUS, nome, data de nascimento, sexo, estado civil, data de cadastro, tipo sanguíneo e se está ativo ou não. Os pacientes podem ter também apelido, CPF, RG, pai e mãe.

Para o médico realizar atendimentos, é necessário para seu cadastro no sistema data de cadastro, a data de início do contrato, o código do conselho, as especialidades que ele atende destacando a principal, o nome, a data de nascimento, o sexo, o estado civil, o tipo sanguíneo e se está ativo ou não. Os médicos podem ter também apelido, CPF, RG, pai e mãe. As especialidades médicas tem que estar previamente cadastradas no sistema com a descrição, a data de cadastro e se está ou não ativa.

Esses atendimentos são realizados nas unidades de atendimento tendo sempre um diagnóstico, a data em que foi realizado, a descrição do atendimento e a indicação se é ou não um retorno. Os atendimentos podem ter também um atendimento pai, e cada atendimento pode estar associado a uma ou várias prescrições. As unidades de atendimento possuem nome e CEP.

O diagnóstico necessita da descrição e dos CIDs que se aplicam a situação do paciente. Cada CID possui um código único mundialmente, a descrição e pode ou não ter um protocolo cadastrado. Um protocolo pode conter vários itens de prescrição para serem sugeridos ao médico no momento em que o mesmo for criar a prescrição para determinado atendimento.

Cada prescrição está associada a um único atendimento, e tem que conter a data em que foi criada. Cada prescrição pode ter também uma descrição e vários itens de prescrição. Os itens de prescrição podem estar ativos ou não e podem ser de três tipos: medicamento, exame e procedimento. Os medicamentos são cadastrados como nome e a substância, e pode ou não estar ativo.

Os exames são cadastrados com o nome e com as orientações em relação ao mesmo. Os procedimentos são cadastrados com o nome e podem ter outro procedimento como pré-requisito.

4.5.5 Estudo de caso 5: Salão de Beleza

Este estudo de caso contempla as características de dados de uma empresa de salão de beleza.

O salão oferece serviços para os clientes. Os responsáveis por vender os serviços são os funcionários que trabalham no salão. O salão também possui diferentes produtos que são fornecidos pelas empresas de fornecimento.

O salão possuirá como características os seguintes atributos: CNPJ, nome, nome do dono, endereço e telefone. Seus funcionários terão como atributos CPF, nome e função. Cada função do funcionário possui um código único. Os produtos do salão possuem descrição, quantidade e data de validade. As empresas fornecedoras possuem CNPJ, nome e produto. No momento em que o fornecedor disponibiliza o produto para o salão, deve ficar registrado qual foi o produto fornecido, a sua data de entrega e o seu valor.

Os clientes do salão devem possuir um cadastro com CPF, nome, endereço e telefone. O funcionário realiza a compra do serviço para o cliente, sendo que serviço deve conter uma descrição e um valor e no momento da compra deve ficar registrado a data da compra e a forma de pagamento.

4.5.6 Estudo de caso 6: Sistema de Imobiliária

Este estudo tem como finalidade armazenar os dados envolvidos no objetivo principal do negócio de qualquer empresa que preste o serviço de encontrar o melhor local para moradia de acordo com o perfil e necessidades do cliente (o inquilino ou locatário) e as expectativas do dono do imóvel (proprietário ou locador). No cenário deste projeto a imobiliária não fornece apenas imóveis para locação, mas sim proporciona ajuda ao cliente durante todas as etapas do processo, desde a escolha do imóvel até a devolução das chaves no término do contrato.

Sendo assim, o cenário que abordaremos neste estudo se baseia em uma pequena imobiliária com atuação reduzida a apenas um bairro de uma cidade. Esta imobiliária é responsável pela locação de apartamentos, escritórios para profissionais autônomos, flats, casas, fazendas, pousadas, chácaras, galpões, armazéns, sítios e outros tipos de imóveis. Ou seja, o banco de dados deve armazenar dados associados a estes tipos de imóvel incluindo características como localização, metragem, quantidade de cômodos, áreas comuns, se possui vaga para estacionamento, etc. Também é preciso guardar informações sobre o histórico de clientes que já alugaram o imóvel e o estado do imóvel antes e depois da locação.

Um dos diferenciais da imobiliária é fornecer uma consultoria de habitação para auxiliar o cliente a encontrar exatamente o que ele está procurando de acordo com o seu perfil. Por isso, é preciso contar com um cadastro de cliente que descreva os principais atributos do cliente e suas necessidades, tais como: moradia temporária (em anos, meses ou final de semana), preferência de localização, locação para um evento com data de início e término e outros. Estes dados são importantes para facilitar a busca e a indicação de um imóvel que o corretor fará durante uma conversa com um cliente em potencial.

Uma vez que o imóvel adequado tenha sido encontrado, o corretor deve ser responsável por diversos aspectos da locação que incluem os detalhes do contrato (incluindo cobrança do aluguel, atrasos e reajustes), do pagamento das contas relacionadas ao imóvel (água, luz, gás, telefonia, internet, etc.) e também da entrega e devolução das chaves. Um requisito importante que precisa ser contemplado é que o imóvel deve ser devolvido no final do período de locação nas mesmas condições que ele estava quando as chaves foram entregues, pois caso contrário, é preciso aplicar uma multa para o inquilino.

Por fim, a imobiliária fornece um serviço diferencial: ela contém um cadastro de profissionais que podem ajudar o cliente nos diversos momentos envolvidos na locação. Por exemplo: se o cliente precisar de algum tipo de manutenção no imóvel como alvenaria, eletricista, encanador, chaveiro, pintor, limpeza (diarista), decorador ou outros, a imobiliária conta com uma lista de profissionais credenciados que auxiliam o inquilino mediante o pagamento do serviço. Também há assistência jurídica e indicação de profissionais de assistência técnica de eletrodomésticos, eletroeletrônicos e de informática. Em resumo, o modelo de dados deve conter cadastros de clientes, imóveis, serviços e outras entidades que serão a base para a criação do modelo de dados.

É muito comum que as pessoas associem a imagem da imobiliária como o papel daquele cobrador que só aparece quando o aluguel está atrasado. Contudo, o papel de uma imobiliária deve ir muito além e sempre estar presente para auxiliar o inquilino e o proprietário em qualquer necessidade de habitação que eles tenham ou possam vir a ter.

4.5.7 Estudo de caso 7: Projeto de Bancos de Dados para Reclamações

Modelar os contatos de consumidores com a empresa

Toda empresa que vende um produto ou presta um serviço deve possuir algum canal de comunicação com seus consumidores para ouvir reclamações. Este canal de comunicação muitas vezes é conhecido como S.A.C., muito utilizado em empresas de médio e grande porte. Outro canal de comunicação muito utilizado é a seção de um site chamada Fale Conosco. De qualquer forma é importante contar com uma maneira de ouvir reclamações, sugestões e comentários gerais do consumidor sobre um produto ou serviço. Com base neste contexto, este projeto tem como objetivo modelar um banco de dados que armazenará os dados envolvidos durante o contato de um consumidor com a empresa.

Ouvindo o consumidor

Cada vez que um consumidor entra em contato com uma empresa para reclamar de um produto ou serviço ele quer ser ouvido e quer obter uma solução satisfatória para o seu problema o mais rápido possível. A maneira na qual o consumidor é atendido durante o período de pós-venda gera vários impactos na empresa, pois caso ele tenha um ótimo atendimento ele não só vai se sentir satisfeito como vai divulgar este fato para outros possíveis consumidores.

Caso contrário, o consumidor terá uma péssima experiência de consumo e dificilmente voltará a adquirir algum produto ou serviço da empresa em questão. Em outras palavras, o que está em jogo durante o processo de atendimento é a credibilidade que a empresa passa aos seus consumidores em relação aos problemas encontrados em seus produtos e/ou serviços.

Devido a esta importante tarefa de ouvir o consumidor, é comum encontrar soluções de C.R.M. que auxiliem todo o processo de relacionamento com o cliente. A importância de um atendimento adequado para reclamações é significativa, pois existem regulamentos e leis que a empresa deve seguir de acordo com órgãos de defesa do consumidor. Além disso, estudos recentes sugerem que cada consumidor mal atendido provavelmente influenciará negativamente pelo menos quatro novos clientes que deixarão de comprar produtos da empresa.

Para simplificar, neste projeto objetiva modelar uma estrutura de armazenamento de dados para uma empresa fictícia chamada ACME. Esta empresa vende diversos produtos, desde patinetes motorizados até armadilhas para caça de animais silvestres velozes. Neste cenário a empresa conta com um S.A.C. onde o consumidor fala pelo telefone com atendentes responsáveis por anotar as reclamações a respeito dos seus produtos. Além do S.A.C. a empresa possui um F.A.Q. em seu site para esclarecer perguntas comuns. O site possui também uma sessão chamada Fale Conosco, onde o consumidor preenche um formulário simples para entrar em contato com a empresa e fazer a sua reclamação. Além destes canais de comunicação, o consumidor também pode ir diretamente a uma das lojas ACME para efetuar sua reclamação do produto comprado.

Do ponto de vista de sistema e de banco de dados é preciso armazenar as informações cadastrais do consumidor. Também é preciso armazenar todos os detalhes da reclamação com informações específicas a respeito do produto, o problema encontrado, data da compra, se o produto está ou não na garantia, a situação que o problema ocorreu, circunstâncias e efeitos colaterais produzidos.

A partir do armazenamento dos dados provenientes da reclamação é importante que a empresa tenha o comprometimento de fornecer algum retorno para o consumidor, indicando quanto tempo em média será necessário para que sua reclamação seja analisada e quais os procedimentos padrões nesta situação. Dentro da empresa é preciso iniciar algum processo para que seja investigada a reclamação do consumidor de acordo com as normas e políticas de qualidade da empresa. O sistema, e consequentemente o banco de dados, também deve ser capaz de armazenar e gerenciar todo o retorno que foi fornecido ao consumidor de acordo com a reclamação recebida.

S.A.C. (Serviço de Atendimento ao Consumidor): Um S.A.C. é uma central de atendimento composta por estruturas físicas e de pessoal e que tem por objetivo centralizar o recebimento de

ligações telefônicas, distribuindo-as automaticamente aos atendentes e possibilitando o atendimento aos usuários finais, geralmente consumidores de uma determinada empresa. Um S.A.C. geralmente faz uso de posições de atendimento (PAs) onde os operadores contam com terminais de vídeo ou computadores ligados em rede que permitem consultar e efetuar registros das chamadas e dos atendimentos realizados. Também são utilizados softwares que monitoram e/ou gravam as ligações telefônicas e controlam o fluxo das chamadas, fornecendo dados para o melhor gerenciamento dos recursos humanos e tecnológicos.

F.A.Q. (Frequently Asked Questions): FAQ é um acrônimo da expressão inglesa que é traduzida como Perguntas Frequentes. Uma F.A.Q. é uma compilação de perguntas frequentes e as respostas destas perguntas acerca de determinado tema. Geralmente um F.A.Q. é disponibilizado como uma página na internet contida dentro do site de uma empresa.

C.R.M. (Customer Relationship Management): CRM é uma expressão em inglês que pode ser traduzida como Gestão de Relacionamento. Foi criada para definir toda uma classe de ferramentas que automatizam as funções de contato com o cliente. O uso destas ferramentas compreendem sistemas informatizados e fundamentalmente uma mudança de atitude corporativa, que objetiva ajudar as companhias a criar e manter um bom relacionamento com seus clientes.

4.5.8 Estudo de caso 8: Projeto de Bancos de Dados para Controle de acesso

Controlar o acesso de pessoas dentro de uma empresa é uma tarefa importante que deve ser implementada em todos os locais restritos de uma empresa. Por questões de segurança, controlar o acesso de uma pessoa dentro de uma empresa pode se relacionar ao tipo de locais que ela pode ir assim como indicar e controlar horários de entrada, saída, pausas para café e outros períodos de tempo que podem ser controlados. Com base nesta necessidade de controle, este trabalho apresenta um estudo de caso de projeto de banco de dados onde toda a estrutura de armazenamento de acessos a uma empresa será modelada para posterior implementação em um banco de dados relacional.

Controlando o acesso

Controlar o acesso de pessoas a locais físicos envolve vários conceitos que vão desde a autenticação até a autorização, passando por questões relacionadas à segurança física e aos recursos utilizados, como armamentos, documentos confidenciais ou valores monetários. Neste trabalho discutiremos a parte de banco de dados necessária para armazenar as informações geradas por um sistema que controla o acesso de pessoas dentro de uma empresa.

Para facilitar a compreensão do que está sendo modelado vamos supor um cenário onde um sistema de controle de acesso pode ser implementado. Vamos imaginar uma empresa cujo ramo de atuação é o transporte de valores. Devido a este ramo de atuação, a empresa precisa saber com exatidão todas as pessoas que possuem acesso a determinadas áreas críticas, como a tesouraria ou a sala que contém armamentos.

Além disso, a empresa também precisa implementar o controle de entrada e saída de funcionários para integrar estes dados com o sistema de RH da empresa, pois é a partir dos registros de entrada e saída que a folha de pagamento vai ser elaborada. De um modo geral, este cenário apresenta muitas características de diversas empresas que precisam saber quando, porquê e como foi feito um determinado acesso a um local da empresa.

Como o objetivo deste trabalho é focar na modelagem, não faz diferença se a empresa utiliza um relógio de ponto eletrônico, uma catraca na portaria ou um leitor de impressão digital biométrico em cada local para controlar o acesso. O que desejamos modelar a partir deste cenário é a estrutura de armazenamento de dados de acordo com os requisitos e as regras de negócio que são geralmente encontradas em um sistema de controle de acesso. Além disso, as questões relacionadas à segurança dos dados também não precisam ser abordadas neste projeto.

4.5.9 Estudo de caso 9: Projeto de Banco de Dados para Controle de Frota

Diversas empresas possuem frotas com veículos utilizados por seus funcionários para realizar seus trabalhos.

Atualmente diversas empresas possuem frotas com veículos utilizados por seus funcionários para realizar seus trabalhos. Conforme a empresa vai crescendo novos veículos vão sendo adquiridos e colocados à disposição daqueles que precisam deles para transporte, locomoção, entregas, retiradas, viagens e outros fins.

Porém, gerenciar o acesso dos funcionários e colaboradores aos veículos requer um modelo adequado para organizar e controlar o uso dos veículos da frota, pois caso contrário há margem para abusos e uso indevido dos recursos. Além disso, é preciso armazenar informações relacionadas à reserva, estado, manutenções, histórico, eventos relevantes e outras entidades que compõem um modelo de banco de dados utilizado por um sistema de controle de frota.

A partir deste cenário este trabalho tratará de um projeto de modelo de banco de dados que pode ser utilizado por qualquer empresa que possui uma frota de veículos cuja alocação é dinâmica e serve para as necessidades dos funcionários e colaboradores.

Entendendo o controle de frota

Para entender como é a utilização dos veículos da frota em uma empresa é preciso primeiro delimitar o escopo e escolher um cenário típico onde há alocação deste tipo de recurso. Neste trabalho vamos considerar uma empresa de tamanho médio que possui mais de 50 funcionários ou colaboradores. Nestas empresas é comum encontrar uma frota de veículos que é disponibilizada para quem precisa deste tipo de recurso com uma condição: o uso do veículo deve ser estritamente voltado para a atividade profissional, ou seja, não é permitido o uso de veículos para fins pessoais.

Neste contexto podemos considerar que a frota é composta de diversos tipos de veículos, como carros, motocicletas, vans, triciclos, tratores, barcos, aviões de pequeno porte, etc. O tipo de veículo em questão deve ser armazenado no nosso modelo, assim como as diversas características próprias do mesmo como, por exemplo, fabricante, modelo, cor, placa, tipo de combustível, quantidade de portas e opcionais. Como existem muitos tipos de opcionais, acessório, customizações e outras características que estão relacionados ao veículo o modelo de banco de dados deve possuir uma maneira de permitir que um usuário administrador do sistema inclua características e valores para estas características dinamicamente. Além disso, toda a documentação e detalhes do estado do veículo também devem ser armazenados. Informações adicionais sobre seguro, tipo de habilitação necessária para conduzi-lo e restrições de uso (como rodízio de acordo com o dia da semana) também devem ser contempladas pelo modelo.

A utilização do veículo para fins profissionais é dividida em duas modalidades: a retirada ocasional, onde é preciso uma reserva e autorização, e o uso como benefício, onde um veículo é alocado permanentemente para um funcionário do alto escalão da empresa.

A modalidade que aloca o veículo permanentemente para um funcionário é caracterizada como um benefício concedido a certos cargos dentro da empresa. Como exemplo pode-se citar a alocação de carros para o presidente, membros do conselho diretor, vice-presidentes e outros cargos. Além de ser um benefício fornecido pelo cargo este privilégio é uma opção estratégica, pois certos profissionais geralmente requerem autonomia e prontidão para se locomover em prol da empresa. Há também o benefício econômico agregado, pois caso não houvesse um veículo alocado permanentemente para certos funcionários seria necessário utilizar outros recursos como um contrato com uma companhia de táxi, por exemplo. Outros detalhes relevantes para o modelo que envolve esta modalidade incluem a alocação de um motorista exclusivo para este veículo, o direito de uso no final de semana e a identificação de um local específico na empresa para o estacionamento. Outro fator importante é a prioridade: caso algum portador deste benefício não esteja com o seu veículo alocado por algum

motivo, a eventual reserva de outro veículo deve ser atendida antes das demais.

O uso do veículo pelos demais funcionários ou colaboradores da empresa requer uma requisição de reserva prévia e a autorização de um funcionário superior a quem está solicitando o recurso. Esta solicitação assume a forma de uma planilha de requisição que deve ser preenchida com antecedência e conter a data de uso, o local de destino, a quilometragem e quantidade de combustível estimada, o local de estacionamento, quem é o supervisor do solicitante, a assinatura de aprovação do supervisor, qual a finalidade de uso do veículo (transporte de carga, viagem, visita ao cliente, retirada de material, etc.), a preferência por algum modelo específico ou acessório, (quatro portas, ar condicionado, etc.) e a solicitação de itens opcionais que não fazem parte do veículo, como capacete, GPS, ou equipamento de segurança. A tarefa de alocar um veículo para o funcionário pode ser manual ou automática, porém os detalhes de como esta alocação é realizada não são relevantes para o modelo, que apenas armazenará a requisição de reserva e o retorno do veículo.

Após a autorização de um superior e do uso do veículo o funcionário/colaborador deve preencher uma planilha de devolução que contém a identificação do funcionário e do veículo, a data/local da retirada e diversas questões sobre o estado do veículo, que incluem a quilometragem percorrida, a quantidade de combustível, o estado da lataria, pneus, motor e outros componentes. Há também nesta planilha um espaço para indicar sugestões e observações, como a necessidade de troca de um componente, manutenção, limpeza, recomendação de uso e aviso sobre problemas futuros. O funcionário também deve indicar a ocorrência de algum evento relevante durante o uso do veículo, que é explicado em seguida.

O acesso ao veículo pode se tornar burocrático demais em determinadas situações emergenciais devido ao controle apresentado pelo modelo e ao fluxo requisição à aprovação à uso à devolução necessário cada vez que algum item da frota for alocado. Sendo assim, o modelo deve considerar o requisito que indica o uso emergencial e simplificar a necessidade de informações necessárias nestas situações. Por exemplo: se houve um acidente na empresa e for preciso levar às pressas um acidentado(a) para o hospital o acesso ao veículo deve ser emergencial e não é preciso seguir o processo comum utilizado nas demais situações. Entretanto, este acesso emergencial deve ser raro e utilizado apenas em situações específicas. Mesmo no acesso emergencial o responsável pela retirada do veículo deve preencher as planilhas de retirada e devolução após o uso e informar que a retirada foi devido a uma emergência.

Diversos eventos relevantes acontecem durante a vida útil dos veículos da frota. Estes eventos incluem manutenções que ocorrem durante o uso, como revisão de peças, lavagem, troca de óleo e pneus e conserto. Além disso, o modelo também deve contemplar eventos como batidas, o uso de guincho, roubos, apreensão, inspeção, lacração e licenciamento. Cada evento relevante para a vida útil do veículo deve possuir um responsável, data, local, comprovante, valor, e outros dados relevantes que devem ser armazenados. Da mesma forma que existem diversas características para um veículo, também há diversos eventos relevantes para o mesmo. Sendo assim, o modelo também deve permitir que um tipo específico de usuário inclua eventos dinamicamente.

4.5.10 Estudo de caso 10: Sistema de Controle de Estoque

Para este estudo de caso iremos nos basear nas necessidades de nosso cliente.

O cliente realizou algumas exigências, sendo elas:

- O sistema deve ter a capacidade de armazenar os produtos contidos no estoque, para que esses possam ser controlados individualmente. Outro detalhe importante no cadastro do produto é armazenar a quantidade mínima que deverá ter desse produto no estoque.
- Cada produto terá um fornecedor relacionado a ele, sendo possível controlar os produtos divididos por fornecedores.
- Os produtos devem ser divididos por categoria, ou seja, cada produto terá uma categoria.
- As entradas e saídas dos produtos deverão ser registradas no programa, para futuramente obtermos um histórico completo de todo o trajeto do produto dentro do centro de distribuição.
- Na entrada do produto será necessário armazenar a data do pedido e a data de entrega da mercadoria, para depois podermos analisar quanto tempo o pedido demora a chegar ao estoque.
- Na saída, obrigatoriamente será informada a loja para a qual a mercadoria foi enviada, pois ao final do mês devemos fazer o fechamento do faturamento para saber qual é a loja que mais obteve vendas.
- Outra capacidade que o sistema deverá ter é calcular o peso total de uma entrada ou de uma saída.
- No programa, devem-se apresentar os produtos nos quais a sua quantidade total em estoque é menor ou igual à quantidade mínima requerida em estoque definida previamente.
- A transportadora será outro item importante na análise, pois devemos saber qual transportadora é mais utilizada para fazer a entrega dos produtos e qual é a mais utilizada para fazer a saída.
- Uma questão que o cliente deseja observar é em qual categoria possui mais item no local.

4.5.11 Estudo de caso 11: Sistema de Almoxarifado

O almoxarifado pertence a um grupo de empresas do ramo industrial e serve para estocar peças destinadas às várias empresas do grupo. Cada empresa do grupo é considerada um cliente do almoxarifado.

O almoxarifado está organizado em corredores. Cada corredor possui vários receptáculos para peças (um receptáculo é uma bacia retangular de material plástico). Os receptáculos são todos do mesmo tamanho. Os corredores são numerados os receptáculos são numerados por corredor. Por exemplo, o receptáculo 2-10 é o décimo receptáculo do segundo corredor.

Em uma das extremidades do almoxarifado encontra-se o setor de recepção de peças. Lá chegam as peças entregues pelos fornecedores. Quando ocorre a chegada de peças, a primeira atividade é registrar na ordem de compra a chegada das peças. Uma cópia de toda ordem de compra é sempre enviada ao setor de recepção. Assim, neste setor sempre sabe-se quais as peças que estão por ser entregues. As ordens de compra são geradas nmo setor de compras e apenas repassadas ao almoxarifado.

Um entrega corresponde sempre a uma ordem de compra. Entretanto, são admitidas entregas parciais, isto é, entregas que não completam a ordem de compra. Em uma entrega podem ser entregues diferentes quantidades de diferentes peças.

As peças recebidas são colocadas sobre um estrado. Este estrado é então levado para o almoxarifado por uma empilhadeira e as peças são distribuídas nos receptáculos. Um estrado pode conter diferentes peças. Para cada peça, procura-se um receptáculo que já contenha unidades da peça em questão e que ainda tenha espaço para a carga chegada. Caso não haja um receptáculo nestas condições, procura-se um receptáculo vazio.

A saída do almoxarifado se dá contra pedidos de clientes. Um pedido pode solicitar vários tipos de peças. Todas as peças que atendem um pedido são juntadas, embaladas e colocadas em uma rampa de carga (numerada) onde encosta o caminhão do cliente. Não há pedidos pendentes, isto é, os clientes sempre pedem quantidades de peças que há em estoque.

O objetivo do sistema é aumentar o lucro do almoxarifado, ajudando sua equipe a guardar e recuperar itens mais rapidamente e a conhecer as quantidades estocadas.

O almoxarifado é de grande porte e constantemente há várias empilhadeiras circulando por ele tanto para estocar entregas quanto para buscar peças referentes a um pedido. Outros detalhes do sistema são fornecidos a seguir.

O almoxarifado somente atende empresas. É necessário manter um cadastro de clientes com CNPJ, nome, endereço e telefone de contato. Para cada peça é necessário conhecer ser UPC (Universal Product Code), descrição e número interno à organização.

Para cada entrega, o setor de recepção monta uma lista de distribuição, que instrui o operador sobre que peças, em que quantidade ele deve estocar em que receptáculos.

Para cada pedido, o setor de saída monta uma lista de busca, que instrui o operador sobre que peças, em que quantidade ele deve buscar em que receptáculos.

Em termos de processos, é necessário que o sistema:

- Dê as ordens de distribuição de peças chegadas para cada chegada.
 - De as ordens para busca para cada pedido.
 - Mantenha a quantidade estocada de cada item e de cada receptáculo.
 - Informe que peças em que quantidade devem ser estocadas ou buscadas em que receptáculos.
- Em termos específicos de transações devem ser consideradas:

- Transação de chegada.
- Registro da chegada de produtos.
- Instruções para estocagem (em que estado, em que receptáculo)
- Confirmação da estocagem em um receptáculo
- Transações de saída de produtos
- Registro de um pedido
- Geração da lista de busca
- Confirmação da busca
- Consolidação de receptáculos (juntar as peças de mesmo tipo de dois receptáculos diferentes).
- Em sua primeira versão, o sistema ainda não fará o controle de empilhaderias ou estrados disponíveis e em uso.

5 Modelo Relacional

5.1 Introdução

O Modelo Relacional estabeleceu-se como um dos primeiros modelos de dados para aplicações comerciais.

O MR foi proposto por CODD (1970) e, entre os modelos de dados de **implementação**, é o mais simples, com estrutura de dados uniforme e mais formal.

O modelo usa o conceito de relação matemática - que se parece com uma tabela de valores - como seu bloco de montagem básico, e sua base teórica reside em uma teoria de conjunto e lógica de predicado de primeira ordem.

5.2 Conceitos do modelo relacional

O modelo relacional representa o banco de dados como uma coleção de *relações*. Informalmente, cada relação é semelhante a uma tabela de valores ou até certo ponto, a um arquivo *plano* de registros. Ele é chamado de arquivo plano porque cada registro tem uma simples estrutura linear ou plana.

Quando uma relação é considerada uma tabela de valores, cada linha na tabela representa uma coleção de valores de dados relacionados. Uma linha representa um fato que normalmente corresponde a uma entidade ou relacionamento do mundo real. Os nomes da tabela e de coluna são usados para ajudar a interpretar o significado dos valores em cada linha. Por exemplo, a primeira tabela da Figura 5.1 é chamada ALUNO porque cada linha representa fatos sobre uma entidade particular de aluno. Os nomes de coluna - Nome, Numero_aluno, Tipo_aluno e Curso - especificam como interpretar os valores de dados em cada linha, com base na coluna em que cada valor se encontra. Todos os valores em uma coluna são do mesmo tipo de dado.

ALUNO			
Nome	Numero_aluno	Tipo_aluno	Curso
Silva	17	1	CC
Braga	8	2	CC

DISCIPLINA			
Nome_disciplina	Numero_disciplina	Creditos	Departamento
Introd. à ciência da computação	CC1310	4	CC
Estruturas de dados	CC3320	4	CC
Matemática discreta	MAT2410	3	MAT
Banco de dados	CC3380	3	CC

Figura 5.1: Exemplo de (parte) de banco de dados que armazena informações de aluno e disciplina

Na terminologia formal do modelo relacional, uma linha é chamada de *tupla*, um cabeçalho da coluna é chamado de *atributo* e a tabela é chamada de *relação*. O tipo de dado que descreve os tipos de valores que podem aparecer em cada coluna é representado por um **domínio** de valores possíveis.

Domínio é o conjunto de valores que um atributo pode ter. Um **domínio** D é um conjunto de valores atômicos. Com atômico, queremos dizer que cada valor no domínio é indivisível em se

tratando do modelo relacional formal. Um método comum de especificação de um domínio é definir um tipo de dado do qual são retirados os valores de dados que formam o domínio. Exemplos de domínios:

- Numeros_telefone_nacional. O conjunto de números de telefone com dez dígitos válidos no Brasil.
- Cadastro_pessoa_física. O conjunto de números do CPF com onze dígitos.
- Nomes. O conjunto de cadeia de caracteres que representam nomes de pessoas.
- Medias_nota. Possíveis valores para calcular a média das notas; cada um deve ser um número real entre 0 e 10.
- Idades_funcionario. Idades possíveis dos funcionários em uma empresa; cada um deve ser um valor inteiro entre 15 e 80.

Estas são chamadas definições lógicas de domínios. Um tipo de dado ou formato também é especificado para cada domínio. Por exemplo, o tipo de dado para o domínio Numeros_telefone_nacional pode ser declarado como uma sequência de caracteres na forma (dd)dddd-dddd, onde cada d é um dígito numérico (decimal) e os dois primeiros dígitos formam um código de área de telefone válidos. Um **esquema relacional** R, indicado por $R(A_1, A_2, \dots, A_n)$, é composto de um nome de relação R e uma lista de atributos, A_1, A_2, \dots, A_n . Cada **atributo** A_i é o nome de um papel desempenhado por algum **domínio** D no esquema de relação R. D é chamado de **domínio** de A_i , e indicado por $\text{Dom}(A_i)$. Um esquema de relação é usado para descrever uma relação; R é chamado de nome dessa relação. O **grau** (ou **aridade**) de uma relação é o número de atributos n desse esquema de relação. Uma relação de grau sete, que armazena informações sobre alunos universitários, teria sete atributos descrevendo cada aluno, da seguinte forma:

```
1 ALUNO(Nome, Cpf, Telefone_residencial, Endereco,
2      Telefone_comercial, Idade, Media)
```

Usando o tipo de dados de cada atributo, a definição às vezes é escrita como:

```
1 ALUNO(Nome: string, Cpf: string, Telefone_residencial: string,
2        Endereco: string, Telefone_comercial: string, Idade: integer,
3        Media: real)
```

Uma **Relação** é um subconjunto de um produto cartesiano de uma lista de domínio. Uma relação r do esquema de relação $R(A_1, A_2, \dots, A_n)$, também indicada por $r(R)$, é um conjunto de n tuplas $r = t_1, t_2, \dots, t_m$. Cada n tuplas t é uma lista ordenada de n valores $t = < v_1, v_2, \dots, v_n >$, em que cada valor v_i , $1 \leq i \leq n$, é um elemento de $\text{dom}(A_i)$ ou é um valor especial NULL. Veja na Figura 5.2.

Tupla é o termo utilizado no MR para designar uma linha na relação. Atributo é utilizado para designar uma coluna na relação.

5.3 Propriedades de uma relação

- **Tuplas são distintas**;
- **Tuplas estão desordenadas**. Uma relação é definida como um conjunto de tuplas. Matematicamente, os elementos de um conjunto não possuem ordem entre eles; logo, as tuplas em

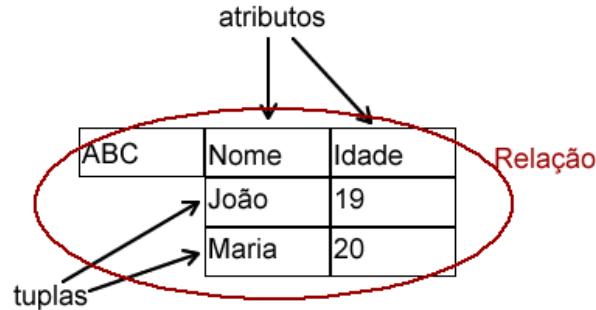


Figura 5.2: Exemplo de atributos e tuplas de uma relação ABC.

uma relação não possuem nenhuma ordem em particular. Em outras palavras, uma relação não é sensível à ordenação das tuplas. Porém, em um arquivo, os registros estão fisicamente ordenados no disco (ou na memória), de modo que sempre existe uma ordem entre eles. Essa ordenação indica o primeiro, segundo, i-ésimo e último registros no arquivo. De modo semelhante, quando exibimos uma relação como uma tabela, as linhas são exibidas em certa ordem. A ordenação da tupla não faz parte da definição da relação porque uma relação tenta representar fatos em um nível lógico ou abstrato. Quando uma relação é implementada como um arquivo ou exibida como uma tabela, uma ordenação em particular pode ser especificada sobre os registros do arquivo ou das linhas da tabela.

- **Todos os valores de atributos são atômicos**, ou seja, ele não é divisível em componentes dentro da estrutura do modelo relacional básico. Logo, atributos compostos ou multivalorados não são permitidos.

5.4 Notação

- uma relação esquema R de grau n é representada por $R(A_1, \dots, A_n)$
- uma tupla é representada como $t = < V_1, V_2, \dots, V_n >$
 - $t[A_i]$ - valor do atributo A_i
 - $t[A_u, A_w, \dots, A_z]$, onde A_u, A_w, \dots, A_z é uma lista de atributos de R, refere-se a uma subtupla de valores $< v_u, v_w, \dots, v_z >$ de t correspondente aos atributos especificados na lista.

Exemplo: $t[nome] = < \text{“João”} >$

$t[nome, idade] = < \text{“João”, 19} >$

Algumas vezes será usado o nome da relação para qualificar o atributo.

Exemplo: ABD.NOME <RELAÇÃO>. <ATRIBUTO>

5.5 Chave primária x Chave candidata x Chave estrangeira

O conceito básico para identificar tuplas e estabelecer relacionamentos entre tuplas é o de chave.

Uma chave é um identificador único para relação, ou seja, duas tuplas distintas nunca têm o mesmo valor para este identificador. Portanto,

$$t_A[CHAVE] \neq t_B[CHAVE]$$

Uma chave deve ser mínima, ou seja, todos os atributos que a compõem são necessários para garantir a unicidade das tuplas.

Uma relação possui apenas uma chave primária, porém pode ter várias chaves candidatas.

Exemplo: Empregado (RG, CPF, matrícula, nome, salário)

$$\text{Chaves candidatas} \left\{ \begin{array}{l} \rightarrow RG \\ \rightarrow CPF \\ \rightarrow MATRICULA \\ MATRICULA + RG + CPF (\text{para ser chave deve ser mínima}) \\ RG + CPF + MATRICULA (\text{para ser chave deve ser mínima}) \end{array} \right.$$

→ RG, CPF e matrícula são chaves candidatas; qualquer uma das três pode ser chave primária.

Uma chave estrangeira é caracterizada por um atributo na relação e é chave primária em outra (ou mesma) relação. Assim é feito o relacionamento. Exemplo: Departamento (sigla, nome)

Empregado (RG, CPF, nome, salário, sigla)

5.6 Restrições do modelo relacional

Restrições sobre os dados que podem ser especificadas em um banco de dados relacional na forma de restrições:

Restrição de Domínio: Especifica que o valor de um atributo deve obedecer a definição de valores admitidos para o atributo.

Restrição de Chave: Define que os valores da chave primária deve ser única.

Restrição de integridade de entidade: Estabelece que nenhum valor da chave primária pode ser nulo.

Restrição de integridade referencial: Estabelece que uma tupla de uma relação que se refere a outra relação deve se referir a uma tupla existente. Lembrar de chave estrangeira.

5.6.1 Outros tipos de restrições

As restrições citadas estão incluídas na linguagem de definição de dados porque ocorrem na maioria das aplicações de banco de dados. No entanto, elas não incluem uma grande classe de restrições gerais, também chamadas de restrições de integridade semântica, que podem ter de ser especificadas e impostas em um banco de dados relacional. Exemplos:

- o salário de um funcionário não deve ser superior ao salário de seu supervisor;
- o número máximo de horas que um funcionário pode trabalhar em todos os projetos por semana é 56.

5.7 Projeto de um banco de dados relacional usando o mapeamento do ER para o relacional

Vamos analisar, agora, como projetar um esquema de um banco de dados relacional tendo por base o esquema de um projeto conceitual. Isso corresponde ao projeto lógico do banco de dados ou ao mapeamento do modelo de dados.

5.7.1 Passos para o mapeamento ER - Relacional

1. Para cada tipo de entidade regular E, crie um esquema de relação R contendo todos os atributos simples de E. Defina a chave primária de R a partir de uma das chaves (identificadores) de E.
2. Para cada tipo de entidade fraca W, crie um esquema de relação R contendo todos os atributos simples de W. Além disso, inclua como atributos de R os atributos que formam a chave primária das relações correspondentes aos tipos de entidade que identificam W. Defina a chave primária de R como a combinação destes atributos e dos atributos derivados da chave parcial de W.
3. Para cada tipo de relacionamento binário R, 1:1, escolha um dos esquemas de relação correspondentes aos tipos de entidades participantes de R, por exemplo, S e inclua como chave estrangeira de S a chave primária do esquema de relação correspondente ao outro tipo de entidade participante de R, definindo-a como chave alternativa de S. Inclua em S os atributos simples de R.
4. Para cada tipo de relacionamento binário R, 1:n, inclua no esquema de relação S correspondente ao tipo de entidade participando do “lado n” de R, como chave estrangeira, a chave primária do esquema de relação correspondente ao tipo de entidade participante do “lado 1” de R. Inclua os atributos de R como atributos de S.
5. Para cada tipo de relacionamento binário R, m:n, crie um esquema de relação S. Inclua como chave estrangeira de S as chaves primárias dos esquemas de relação correspondentes aos tipos de entidade participantes de R. Defina como chave primária de S a combinação das chaves estrangeiras. Além disso, inclua como atributos de S os atributos simples de R.
6. Para cada atributo multivalorado A, crie um esquema de relação R, que inclua um atributo correspondente a A e, como chave estrangeira, a chave primária K do esquema de relação correspondente ao tipo de entidade ou relacionamento que contém A. Defina como chave primária de R a combinação de todos os seus atributos.
7. Para cada tipo de relacionamento n-ário ($n > 2$) R, crie um esquema de relação S. Inclua como chave estrangeira de S as chaves primárias dos esquemas de relação correspondentes aos tipos de entidade participantes de R. Defina a chave primária de S como uma combinação das chaves estrangeiras, de acordo com a relação de cardinalidade de R.

5.7.2 Passos para o mapeamento EER - Relacional

Mapeamento da especialização ou generalização

A etapa 8 que vem a seguir, oferece as opções mais comuns; outros mapeamentos também são possíveis. Usaremos como notação: Atrs(R) para indicar os atributos da relação R e ChP(R) para indicar a chave primária de R. Etapa 8: Opções para mapeamento da especialização ou generalização. Converta cada especialização com m subclasses $\{S_1, S_2, \dots, S_m\}$ e superclasse (generalizada) C, e, que os atributos de C são $\{ch, a_1, \dots, a_n\}$ e ch é a chave (primária) para os esquemas da relação usando uma das seguintes opções:

- **Opção 8A:** Múltiplas relações superclasse e subclasses. Crie uma relação L para C com atributos $Atrs(L) = \{ch, a_1, \dots, a_n\}$ e $ChP(L_i) = ch$. Crie uma relação L_i , para cada subclassse $S_i, 1 \leq i \leq m$, com os atributos $Atrs(L_i) = \{ch\} \cup \{\text{atributos de } S_i\}$ e $ChP(L_i) = ch$. Essa opção funciona para qualquer especialização (total ou parcial, disjunta ou sobreposta).

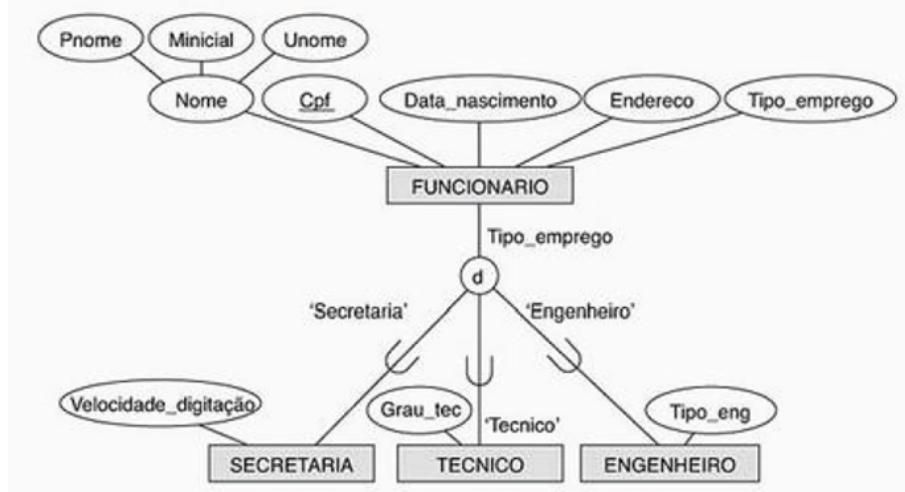


Figura 5.3: Notação do diagrama EER para uma especialização definida por atributo

- **Opção 8B:** Múltiplas relações apenas relações de subclasse. Crie uma relação L_i para cada subclasse $S_i, 1 \leq i \leq m$, com atributos $Atrs(L_i) = \{\text{atributos de } S_i\} \cup \{ch, a_1, \dots, a_n\}$ e $ChP(L_i) = ch$. Essa opção só funciona para uma especialização cujas subclases são totais (cada entidade na superclasse deve pertencer a (pelo menos) uma das subclases). Além disso, isso só é recomendado se a especialização tiver a restrição de disjunção. Se a especialização for sobreposta, a mesma entidade pode ser duplicada em várias relações.

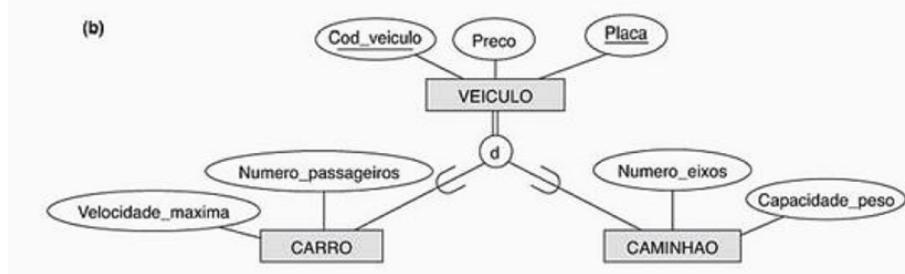


Figura 5.4: Generalização. Generalizando CARRO e CAMINHAO na superclasse VEICULO

- **Opção 8C:** Relação única com um atributo de tipo. Crie uma relação L para C com atributos $Atrs(L) = \{ch, a_1, \dots, a_n\} \cup \{\text{atributos de } S_1\} \cup \dots \cup \{\text{atributos de } S_m\} \cup \{t\}$ e $ChP(L) = ch$. O atributo t é chamado de atributo de tipo (ou discriminador), cujo valor indica a subclasse à qual cada tupla pertence, se houver alguma. Essa opção funciona somente para uma especialização cujas subclases são disjuntas, e tem o potencial para gerar muitos valores NULL se diversos atributos específicos existiram nas subclases.

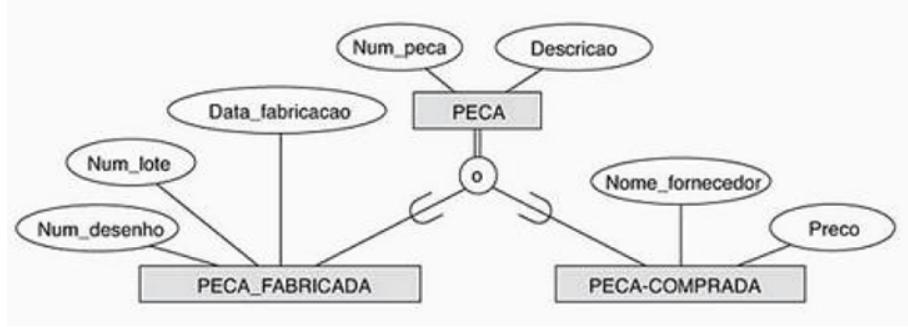


Figura 5.5: Notação de diagrama EER para uma especialização sobreposta (não disjunta)

- **Opção 8D:** Relação isolada com atributos de múltiplos tipos. Crie um único esquema de relação L com atributos $Atrs(L) = \{ch, a_1, \dots, a_n\} \cup \{\text{atributos de } S_1\} \cup \dots \cup \{\text{atributos de } S_m\} \cup \{t_1, t_2, \dots, t_m\}$ e $ChP(L) = ch$. Cada $t_i, 1 \leq i \leq m$, é um atributo de tipo booleano indicando se uma tupla pertence à subclasse S_i . Essa opção é usada para uma especialização cujas subclasses são sobrepostas (mas também funcionará para uma especialização disjunta).

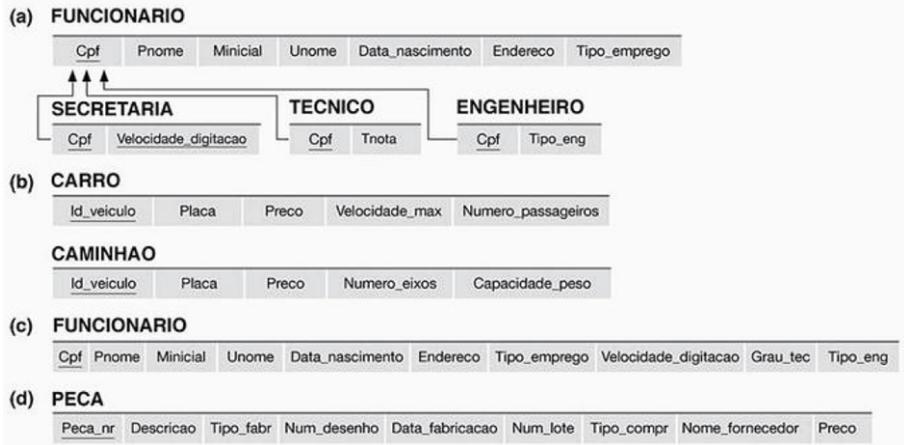


Figura 5.6: Opções para mapeamento de especialização ou generalização. (a) Mapeando o esquema EER na Figura 5.3 ao usar a opção 8A. (b) Mapeando o esquema EER na Figura 5.4 ao usar a opção 8B. (c) Mapeando o esquema EER na Figura 5.3 ao usar a opção 8C. (d) Mapeando a Figura 5.5 ao usar a opção 8D com campos de tipo booleano Tipo_fabr e Tipo_compr

5.8 Exercícios

1. A Figura 5.7 mostra um esquema ER para um banco de dados que pode ser usado para registrar navios de transporte e seus locais para autoridades marítimas. Mapeie esse esquema para um esquema relacional e especifique todas as chaves primárias e estrangeiras.

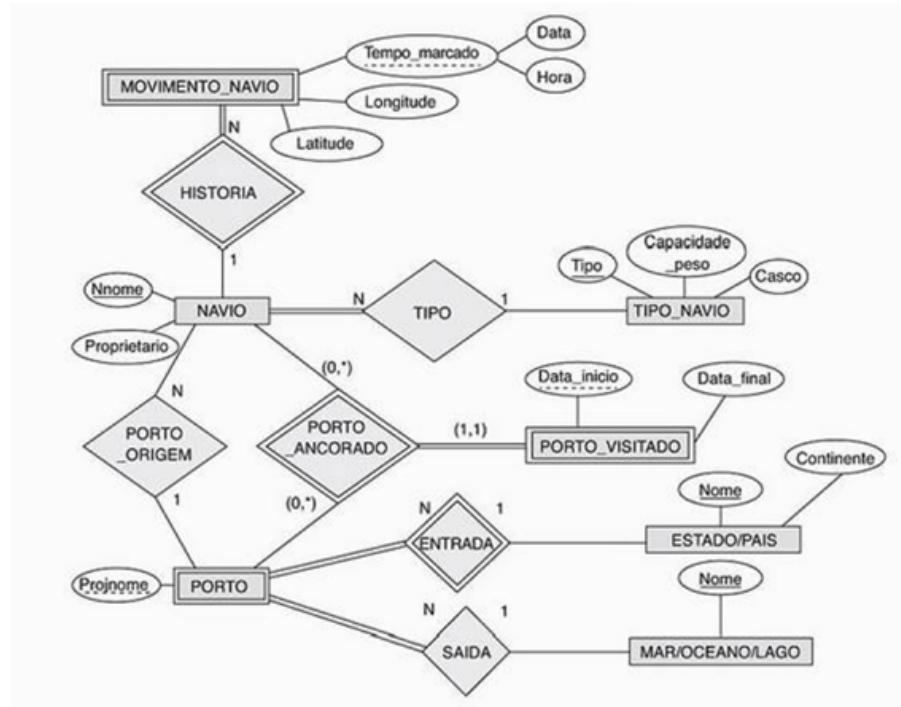


Figura 5.7: Esquema ER para o banco de dados MOVIMENTO_NAVIO

2. Mapeie o esquema ER BANCO da Figura 5.8 em um esquema relacional. Especifique todas as chaves primárias e estrangeiras.

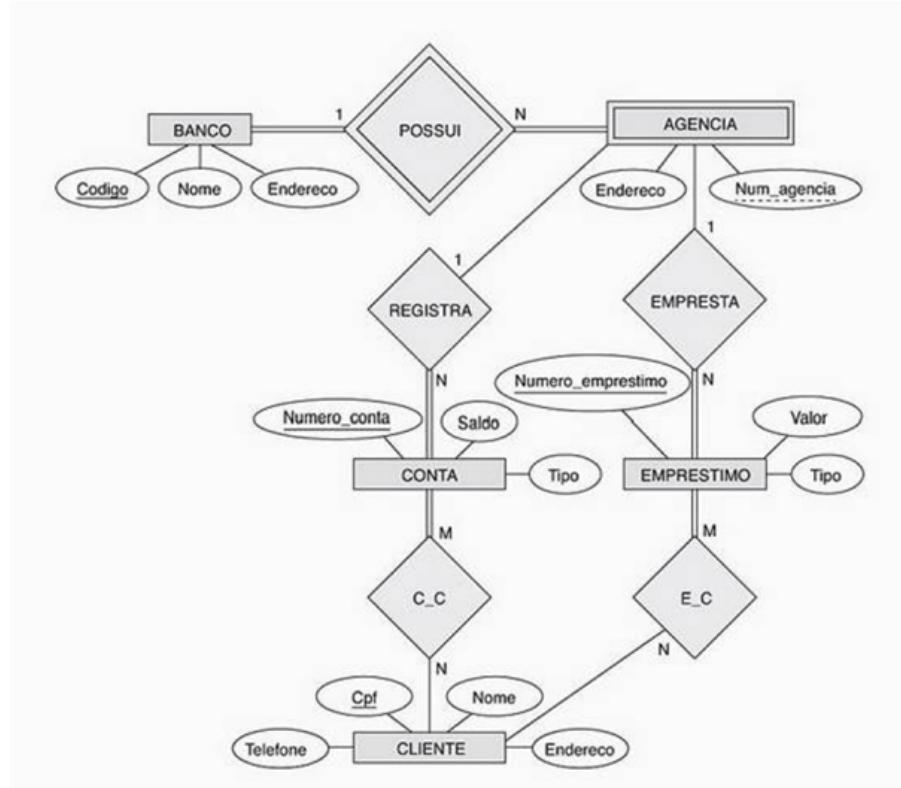


Figura 5.8: Diagrama ER para um esquema de banco de dados BANCO

3. Faça o mapeamento ER para modelo relacional do esquema COMPANHIA_AEREA da Figura 5.9. Especifique todas as chaves primárias e estrangeiras.

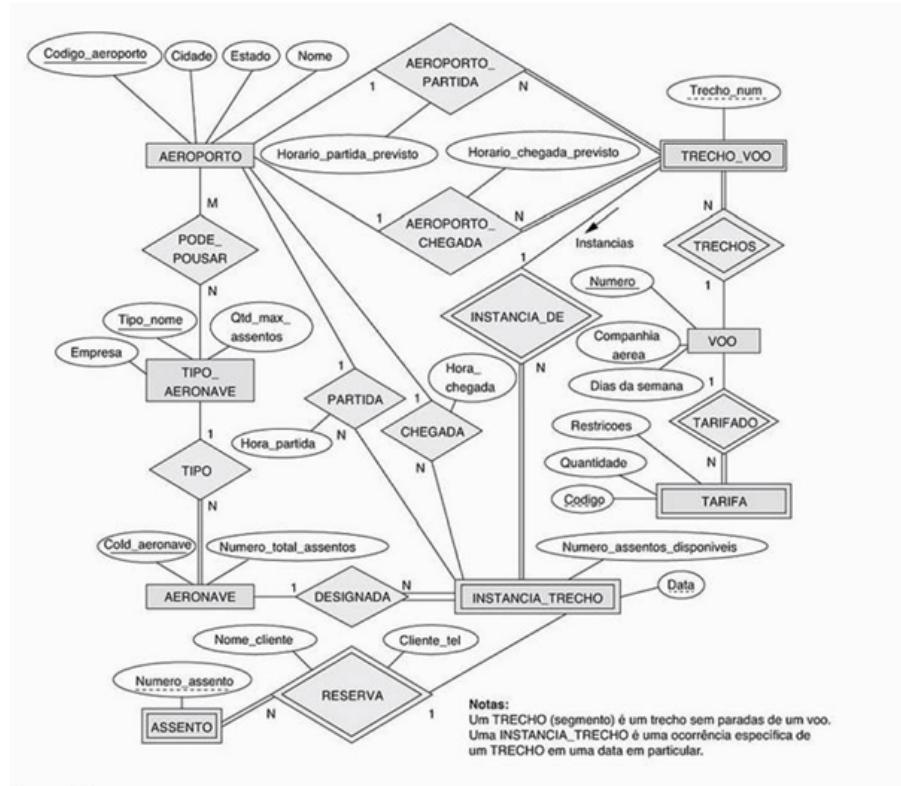


Figura 5.9: Diagrama ER para um esquema de banco de dados COMPANHIA_AEREA

4. Faça o mapeamento do modelo de esquema ER EMPRESA da Figura 5.10 para o modelo relacional. Especifique todas as chaves primárias e estrangeiras.

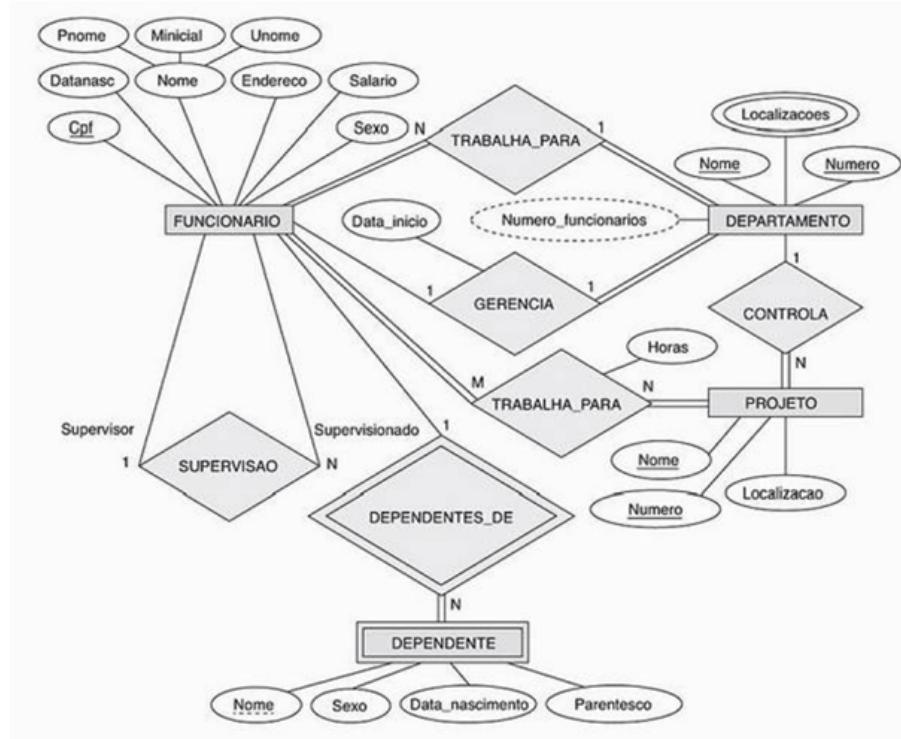


Figura 5.10: Diagrama ER para um esquema de banco de dados EMPRESA

5. Seja o modelo entidade relacionamento da Figura 5.11, que descreve um minimundo simplificado para reservas de hotel. Faça o mapeamento para o Modelo Relacional (modelo de tabelas). Indique claramente as chaves primárias e chaves estrangeiras.

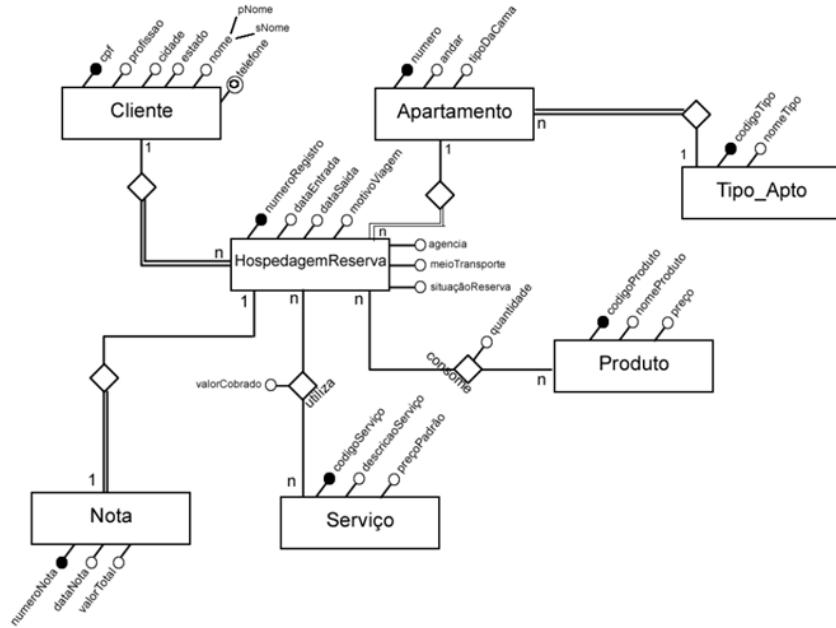


Figura 5.11: Diagrama ER para um banco de dados de reservas de hotel

6. Escreva o esquema relacional para o modelo ER apresentado na Figura 5.12:

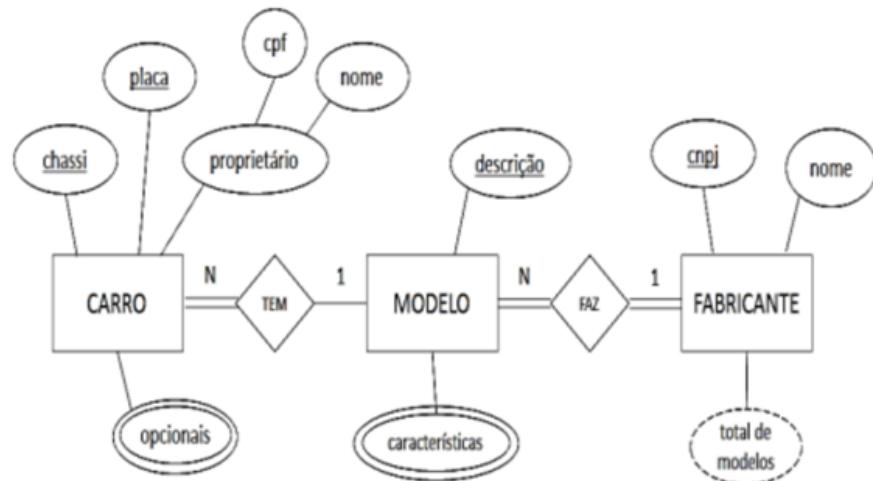


Figura 5.12: Entidade Relacionamento para um sistema de ‘fábrica de carros’

7. Para o esquema relacional abaixo:

Hospital(codigo, nome, endereco, telefone, cidade, uf, CNPJ(FK-Universidade), especialidade)

Universidade(cnpj, nome, cidade)

Podemos afirmar que: (escolha uma ou mais)

- (a) A cardinalidade do relacionamento universidade-hospital é 1:n
- (b) Um Hospital está associado a apenas uma Universidade
- (c) Endereco é um atributo composto na tabela Hospital
- (d) Telefone é um atributo multivalorado.
- (e) Um hospital está associado a uma ou mais Universidades
- (f) Uma Universidade pode ter vários hospitais vinculados a ela.
- (g) O atributo da tabela Hospital que faz o relacionamento com Universidade é o “especialidade”

8. Faça o mapeamento do diagrama EER da Figura 5.13. Nos casos de generalização/especialização use uma das opções de mapeamento apresentadas na aula.

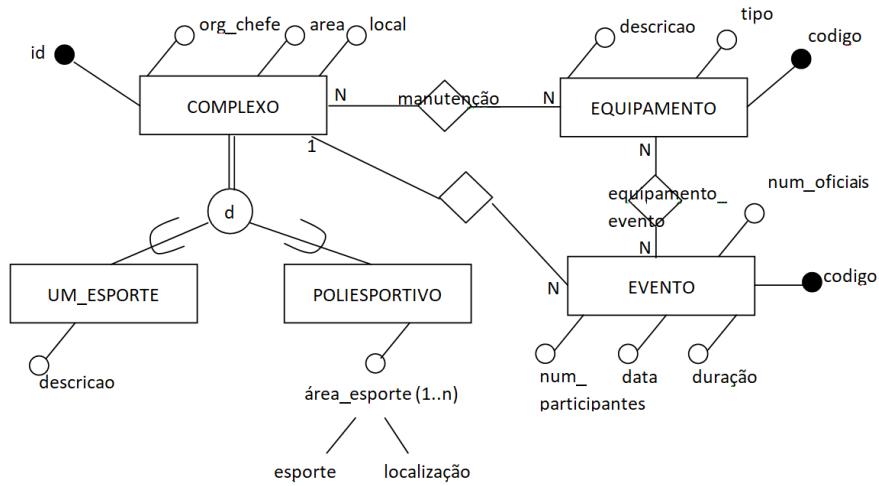


Figura 5.13: Diagrama EER para banco de dados de um complexo poliesportivo

9. Faça o mapeamento do diagrama EER da Figura 5.14, a qual mostra as restrições para um banco de dados de um aeroporto particular. Nos casos de generalização/especialização use uma das opções de mapeamento apresentadas na aula.

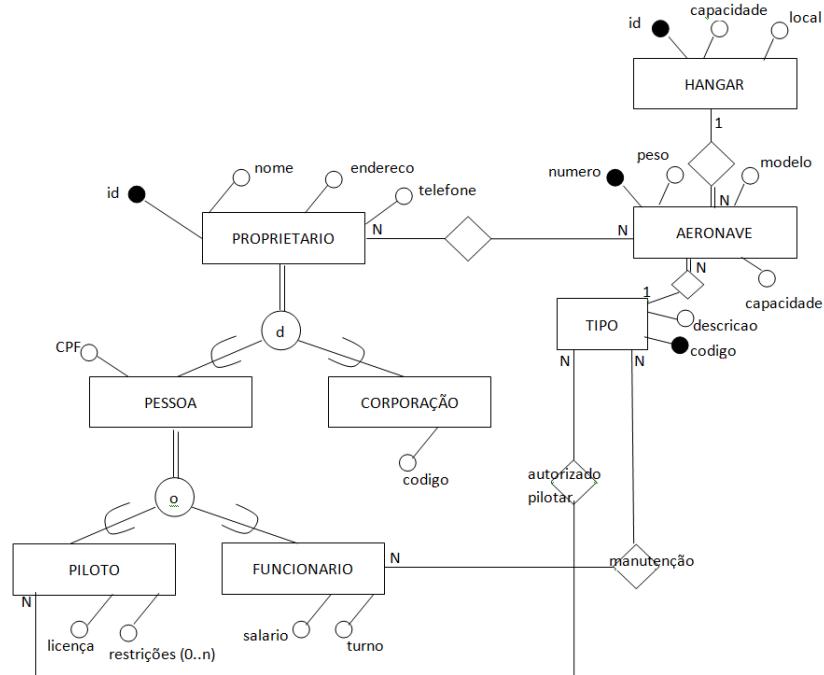


Figura 5.14: Diagrama EER para banco de dados de um aeroporto

6 Álgebra relacional

6.1 Introdução

É uma coleção de operações para manipular as relações. São usadas para selecionar tuplas ou combinar tuplas relacionadas a diversas relações.

As operações são divididas em dois grupos:

- Específica para Banco de Dados Relacional: seleção, projeção, junção e divisão;
- Teoria posconjunto: união, interseção, diferença e produto cartesiano.

6.2 Seleção

Seleciona um subconjunto de tuplas de uma relação que satisfaz uma condição. Sintaxe:

$$RESULTADO \leftarrow \sigma_{<\text{CONDIÇÃO}>}(<\text{RELAÇÃO}>)$$

A operação seleção é unária e comutativa.

$$\begin{aligned} \sigma_{<\text{CONDIÇÃO } 1>}(\sigma_{<\text{CONDIÇÃO } 2>}(<\text{RELAÇÃO}>)) \\ \parallel \\ \sigma_{<\text{CONDIÇÃO } 2>}(\sigma_{<\text{CONDIÇÃO } 1>}(<\text{RELAÇÃO}>)) \end{aligned}$$

Condição é qualquer expressão booleana (expressão lógica).

$$\begin{aligned} \sigma_{<\text{COND } 1>}(\dots(\sigma_{<\text{COND } n>}(<\text{RELAÇÃO}>))\dots) \\ \sigma_{<\text{COND } 1>} AND \dots AND <\text{COND } n>(<\text{RELAÇÃO}>) \end{aligned}$$

Exemplo: Obter a relação de empregados que trabalham no departamento 4.

$$RESULTADO \leftarrow \sigma_{NDEPT=4}(EMPREGADO)$$

Obter a relação de empregados que tenham salário maior do que 3000.

$$RESULTADO \leftarrow \sigma_{SALARIO>3000}(EMPREGADO)$$

Obter a relação de empregados que trabalham no departamento 4 e ganham mais de 2500 ou no departamento 5 e ganham menos do que 3000.

$$RESULTADO \leftarrow \sigma_{((NDEPT=4 \text{ AND } SALARIO > 2500) \text{ OR } (NDEPT=5 \text{ AND } SALARIO < 3000))}(EMPREGADO)$$

6.3 Projeção

Seleciona um subconjunto de atributos de uma relação.

Sintaxe:

$$RESULTADO \leftarrow \pi_{<\text{atributos}>}(<\text{RELAÇÃO}>)$$

Observação: Elimina as tuplas duplicadas, a relação resultante possui apenas uma cópia.

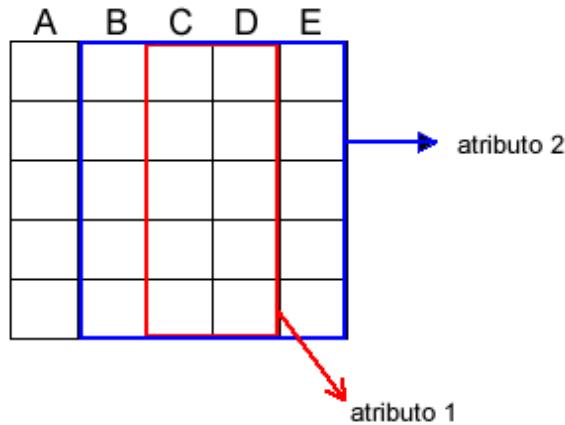
chave	nome	salario
1	José	750
2	Toni	1000
3	Lucas	500
4	Guido	750

$$\pi_{\langle \text{atributo } 1 \rangle}(\pi_{\langle \text{atributo } 2 \rangle}(<\text{RELAÇÃO}>))$$

||

$$\pi_{\langle \text{atributo } 1 \rangle}(<\text{RELAÇÃO}>))$$

Quando $\langle \text{atributo } 2 \rangle$ contém os $\langle \text{atributo } 1 \rangle$.



Exemplo: Obter a relação do primeiro nome dos empregados.

$$RESULTADO \leftarrow \pi_{Pnome}(<\text{EMPREGADO}>)$$

Podem ocorrer combinações. Exemplo:

Obter a relação do primeiro e último nome de todos os empregados que trabalham no departamento 5.

$$RESULTADO \leftarrow \pi_{(Pnome, Unome)}(\sigma_{Ndep=5}(EMPREGADO))$$

A relação abaixo não produz o mesmo resultado:

$$RESULTADO \leftarrow \sigma_{Ndep=5}(\pi_{(Pnome, Unome)}(EMPREGADO)) \Rightarrow ERRO!$$

Para dar o mesmo resultado, poderia fazer:

$$RESULTADO \leftarrow \sigma_{Ndep=5}(\pi_{(Pnome, Unome, Ndep)}(EMPREGADO))$$

6.4 Junção

Combina tuplas relacionadas em uma única tupla.

Sintaxe:

$RESULTADO \leftarrow <\text{RELAÇÃO}> \bowtie <\text{RELAÇÃO}>_{(\text{condição})}$

Seja $R(A_1, \dots, A_n)$ e $S(B_1, \dots, B_M)$, então:

$Q \leftarrow R \bowtie S_{(\text{condição})}$

$Q(A_1, \dots, A_n, B_1, \dots, B_M)$

Equi Join → a condição utilizada na junção é uma igualdade.

Natural Join → elimina o segundo atributo da condição.

Exemplo: Obter a relação do nome dos gerentes e o nome dos departamentos.

$DEPTO_GER \leftarrow DEPARTAMENTO \bowtie EMPREGADO_{(NSSGER=NSS)} \Rightarrow \text{Equi Join}$

$RESULTADO \leftarrow \pi_{(\text{Dnome, Pnome})}(DEPT_GER)$
 $DEPTO_GER$

Dnome	Dnumero	Nssger	...	Pnome	Mnome	Uname	NSS...

$DEPTO_GER \leftarrow DEPARTAMENTO * EMPREGADO_{(NSSGER=NSS)} \Rightarrow \text{Natural Join}$

6.5 União

Efetua a união de duas relações com mesmo grau N e $DOM(A_i) = DOM(B_i), 1 \leq i \leq N$.
 $RESULTADO \leftarrow <\text{RELAÇÃO}> \cup <\text{RELAÇÃO}>$

6.6 Interseção

Efetua a interseção de duas relações com mesmo grau N e $DOM(A_i) = DOM(B_i), 1 \leq i \leq N$.
 $RESULTADO \leftarrow <\text{RELAÇÃO}> \cap <\text{RELAÇÃO}>$

6.7 Diferença

Efetua a diferença de duas relações com mesmo grau N e $DOM(A_i) = DOM(B_i), 1 \leq i \leq N$.
 $RESULTADO \leftarrow <\text{RELAÇÃO}> - <\text{RELAÇÃO}>$

6.8 Produto cartesiano

Combina tuplas de duas relações que precisam ser compatíveis.
 $RESULTADO \leftarrow <\text{RELAÇÃO}> X <\text{RELAÇÃO}>$

Seja:

$R(A_1, \dots, A_n)$

$S(B_1, \dots, B_m)$

$Q \leftarrow R \times S$

$Q(A_1, \dots, A_n, B_1, \dots, B_m)$

Exemplo:

1. Obter a relação do nome dos dependentes dos empregados do sexo feminino.

$$\begin{aligned} Emp_Fem &\leftarrow \sigma_{\text{sexo} = "F"}(Empregado) \\ Emp_Nomes &\leftarrow \pi_{P\text{nome}, U\text{nome}, NSS}(Emp_Fem) \\ Emp_Dep &\leftarrow Emp_Nomes \times \text{Dependentes} \\ Dep_Atual &\leftarrow \sigma_{NSS = NSSEMP}(Emp_Dep) \\ RESULTADO &\leftarrow \pi_{P\text{nome}, S\text{Nome}, Nome\text{Dependente}}(Dep_Atual) \end{aligned}$$

6.9 Divisão

$RESULTADO \leftarrow <REL> \div <REL>$

Exemplo:

Obter a relação dos empregados que trabalham em todos os projetos em que “John Smith” trabalha.

$$\begin{aligned} SMITH &\leftarrow \sigma_{P\text{nome} = "John" \text{ and } S\text{nome} = "Smith"}(Empregado) \\ SMITH_PNO &\leftarrow \pi_{P\text{nro}}(Trabalha_Em * (\sigma_{NSSEMP = NSS}(SMITH))) \\ NSS_PNRO &\leftarrow \pi_{P\text{nro}, nssemp}(Trabalha_Em) \\ NSS_DESEJADO &\leftarrow NSS_PNRO \div SMITH_PNO \\ RESULTADO &\leftarrow \pi_{P\text{nome}, S\text{Nome}}(NSS_DESEJADO * EMPREGADO) \end{aligned}$$

7 SQL - Structured Query Language

7.1 Introdução

É um conjunto de declarações utilizado para acessar os dados em BD. Não é uma linguagem procedural pois processa um conjunto de registros ao invés de um por vez. É utilizada pelo DBA, administrador do sistema, programadores, sistema de tomada de decisão e outros usuários finais. SQL é a linguagem padrão de definição e manipulação para bancos de dados relacionais.

A SQL foi criada com o objetivo de padronizar os comandos de definição e manipulação de dados em SGBD's relacionais. Hoje em dia, apesar de a linguagem possuir uma quantidade considerável de extensões e implementações proprietárias (visto que cada SGBD em participar pode implementar variações sobre seus comandos), pode-se afirmar que a meta foi alcançada.

A SQL é subdividida em linguagens, que são:

DDL (Data Definition Language): que apresenta uma série de comandos que permitem a definição dos dados. Dentre os principais comandos, podemos destacar o CREATE, que é destinado à criação do banco de dados e das tabelas que o compõe, além das relações existentes entre as tabelas. Outros exemplos de comandos DDL são o ALTER e o DROP.

DML (Data Manipulation Language): que apresenta uma série de comandos destinados à manipulação de dados tais como: consultas, inserções, exclusões e alterações, para um ou mais registros de uma ou mais tabelas de maneira simultânea. Dentre alguns exemplos de comandos DML, destacam-se: SELECT, INSERT, UPDATE e DELETE.

DCL (View Definition Language): que apresenta uma série de comandos para controlar o acesso aos dados, usuários e grupos. Dentre alguns exemplos de comandos podemos destacar o comando GRANT e o comando REVOKE. A VDL é também chamada por alguns autores de DCL (Data Control Language).

Para estudarmos os comandos principais da SQL, usaremos como exemplo um esquema de banco de dados de um sistema de controle de vendas de uma loja. A definição do sistema e os modelos conceitual e lógico vem a seguir.

Sistema de controle de vendas de uma loja

Descrição do sistema:

Uma loja pretende controlar seu estoque, suas vendas, clientes e vendedores.

Os **clientes** da loja devem ser registrados com nome, endereço completo (cidade, rua, bairro, etc.) e telefone (que pode ser mais de um).

Cada **produto** da loja deve ser registrado com um nome característico, a descrição do produto, valor unitário e a quantidade em estoque.

Os **vendedores** da loja são registrados com seu nome, salário fixo e taxa de comissão.

As vendas podem corresponder a um ou mais produtos. De forma que o vendedor é o responsável por cadastrar a venda no sistema, informando o vendedor responsável, o cliente que está realizando a compra e os itens de produto (com quantidade informada de cada item), além do prazo de entrega (opcional, caso se trate de entrega realizada pela loja).

7.2 Linguagem de Definição de Dados (DDL)

7.2.1 Criação do esquema - CREATE SCHEMA

O primeiro comando DDL que precisamos detalhar é o comando para a criação do banco de dados, cuja sintaxe padrão é definida no Script 1, onde nome_db é o nome do banco de dados a ser criado.

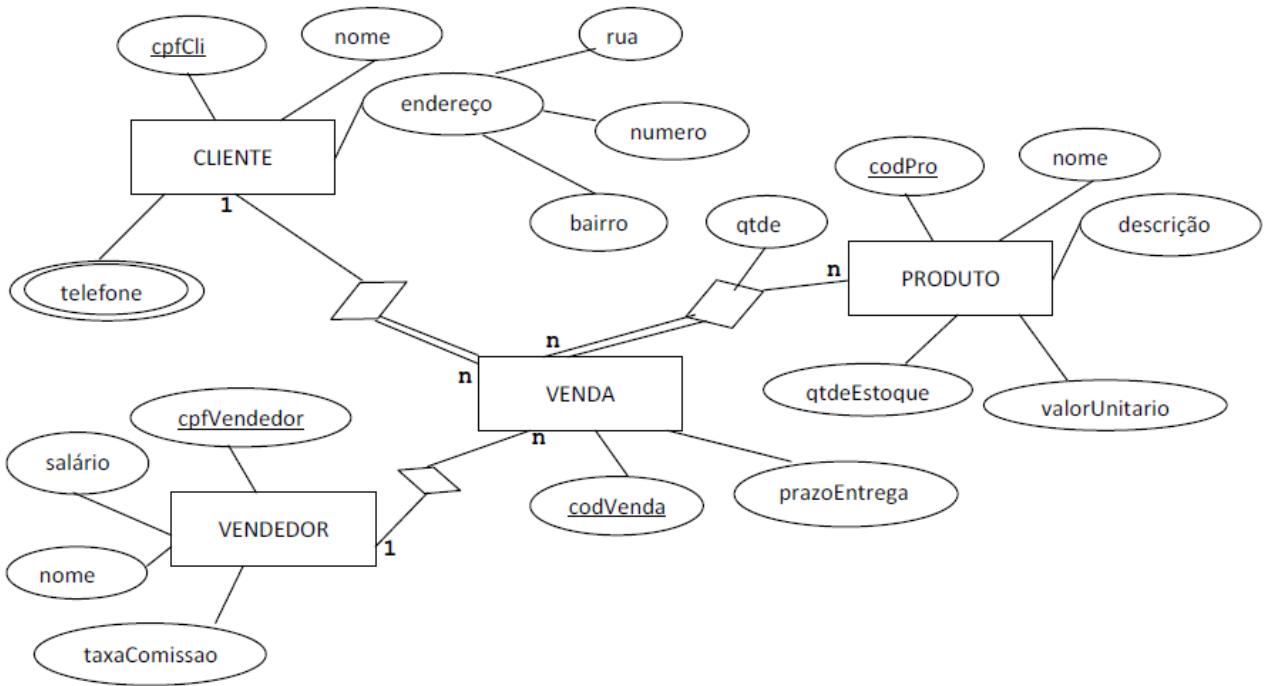


Figura 7.1: Modelo conceitual para o projeto Loja.

Script 1: Criação do banco de dados

```
1 CREATE DATABASE <nome_db>;
```

Para SGBD MySQL, a criação do banco de dados pode ser realizada tanto com o Script 1 quanto com o Script 2.

Script 2: Criação do banco de dados, para MySQL

```
1 CREATE SCHEMA <nome_db>;
```

Após a definição da sintaxe, utilizaremos os comandos SQL para a criação do banco de dados e das tabelas de um determinado modelo de dados. Como modelo de exemplo, utilizaremos o sistema de controle de vendas de uma loja (Veja a descrição acima e os modelos gerados na Figura 7.1 e Figura 7.2).

No Script 5 temos o comando SQL para criar o banco de dados com o nome de loja. Após executar este podemos observar o banco de dados criado.

Script 3: Criação do banco de dados loja

```
1 CREATE DATABASE loja;
```

Outro importante comando DDL é o **CREATE USER**, que cria um usuário do banco de dados. O Script 4 mostra como criar o usuário ‘adm’ no banco de dados. Veja que foi definido o usuário adm para o domínio da máquina local (localhost). Além disso foi definida a senha *123* para este usuário.

Script 4: Criação do usuário adm

```
1 CREATE USER adm@localhost IDENTIFIED BY 123;
```

CLIENTE				
<u>cpfCli</u>	nome	rua	numero	bairro
PRODUTO				
<u>codPro</u>	nome	descricao	valorUnitario	qtdeEstoque
VENDEDOR				
<u>cpfVendedor</u>	nome	salario	taxaComissao	
VENDA				
<u>codVenda</u>	prazoEntrega	<u>cpfVendedor</u>	<u>cpfCli</u>	
PRODUTO_VENDA				
<u>codVenda</u>	<u>codPro</u>	qtde		
TELEFONE				
<u>codCli</u>	telefone			

Figura 7.2: Modelo lógico para o projeto Loja.

No Script ?? temos a concessão de privilégio para o usuário adm no banco de dados loja, no caso específico do MySQL.

Script 5: Privilégio concedido ao usuário ‘adm’

```
1 GRANT ALL PRIVILEGES ON loja.* to adm@localhost;
2 FLUSH PRIVILEGES;
```

O comando FLUSH PRIVILEGES faz um recarregamento (*reload*) na tabela de privilégios do SGBD. Falaremos com mais detalhes sobre a concessão e revogação de privilégios quando estudarmos comandos específicos da VDL.

Se precisarmos excluir um esquema de banco de dados, temos o comando DROP DATABASE ou DROP SCHEMA (Script 6). Este comando exclui o esquema banco de dados e todas as estruturas criadas dentro dele (tabelas, functions, índices, procedures, triggers, etc.). Muito cuidado com o comando DROP SCHEMA.

Script 6: Comando para exclusão do esquema de banco de dados

```
1 DROP SCHEMA <nome_do_bd>;
```

7.2.2 Criação de tabelas - CREATE TABLE

O comando básico para criar tabelas em banco de dados relacionais é o CREATE TABLE, cuja sintaxe básica pode ser vista no Script 7, onde:

Nome_tabela: indica o nome da tabela a ser criada;

Nome_atributo1...Nome_atributoN: são os nomes dos atributos da tabela a ser criada;

Tipo_Dado1 ... Tipo_DadoN: são os tipos de dados de cada um dos atributos da tabela a ser criada;

Atributo_chave: é o nome do atributo chave primária da tabela a ser criada;

Atributo_chave_estrangeira: é o atributo da tabela a ser criada que é chave estrangeira, fazendo referência a outra tabela;

Tabela_referenciada: é a tabela cuja chave estrangeira faz referência;

Atributo_chave_tabela_referenciada: é o nome do atributo chave primária da tabela referenciada.

NOT NULL: é o comando que identifica não pode receber valores nulos. O padrão é NULL.

Script 7: Criação de Tabelas em Banco de Dados

```

1 CREATE TABLE <Nome_tabela> (
2     <Nome_atributo1> <Tipo_Dado1> [NOT NULL] ,
3     <Nome_atributo2> <Tipo_Dado2> [NOT NULL] ,
4     ...
5     <Nome_atributoN> <Tipo_DadoN> [NOT NULL] ,
6     PRIMARY KEY (Atributo_chave),
7     FOREIGN KEY(Atributo_chave_estrangeira)
8         REFERENCES Tabela_referenciada(Atributo_ch_tab_referenciada)
9 );

```

Existem variações dessa sintaxe, como veremos nos nossos exemplos.

No modelo de dados utilizado como exemplo podemos verificar a existência das seguintes tabelas:

- Cliente
- Produto
- Vendedor
- Venda
- Produto_Venda
- Telefone

Criaremos essas tabelas no banco de dados *loja*. Antes, vamos discutir a ordem de criação dessas tabelas. Podemos criá-las partindo das tabelas que não possuem chaves estrangeiras. Essa ordem é conveniente pois, se criarmos uma chave estrangeira nos referindo a uma tabela que ainda não existe, teremos um erro. Existem outras maneiras de corrigir este problema sem nos importarmos com a ordem de criação das tabelas. Discutiremos esse detalhe mais adiante. Por hora, criaremos as tabelas de acordo com a seguinte ordem:

1. Cliente
2. Produto
3. Vendedor
4. Venda
5. Produto_Venda
6. Telefone

Script 8: Criação da tabela cliente no banco de dados loja

```

1 CREATE TABLE loja.cliente (
2     cpfCli INTEGER NOT NULL ,
3     nome VARCHAR(45) NOT NULL ,
4     rua VARCHAR(100) ,
5     numero VARCHAR(5) ,
6     bairro VARCHAR(20) ,
7     PRIMARY KEY (cpfCli)
8 );

```

ou

Script 9: Criação da tabela cliente no banco de dados loja

```

1 CREATE TABLE loja.cliente (
2     cpfCli INTEGER NOT NULL PRIMARY KEY ,
3     nome VARCHAR(45) NOT NULL ,
4     rua VARCHAR(100) ,
5     numero VARCHAR(5) ,
6     bairro VARCHAR(20)
7 );

```

Script 10: Criação das tabelas produto e vendedor

```

1 CREATE TABLE loja.produto (
2     codPro INTEGER NOT NULL AUTO_INCREMENT ,
3     nome VARCHAR(100) NOT NULL ,
4     descricao VARCHAR(200) ,
5     valorUnitario DOUBLE NOT NULL ,
6     qtdeEstoque INTEGER DEFAULT 0 ,
7     PRIMARY KEY (codPro)
8 );
9
10 CREATE TABLE loja.vendedor (
11     cpfVendedor integer not null ,
12     nome varchar(45) not null ,
13     salario DOUBLE NOT NULL DEFAULT 750.00 ,
14     taxaComissao DOUBLE NOT NULL DEFAULT 2.5 ,
15     PRIMARY KEY (cpfVendedor)
16 );

```

Script 11: Criação das tabelas venda, produto_venda e telefone

```

1 CREATE TABLE loja.venda (
2     codVenda INTEGER NOT NULL AUTO_INCREMENT ,
3     cpfVendedor INTEGER NOT NULL ,
4     cpfCli INTEGER NOT NULL ,
5     prazoEntrega DATE ,
6     PRIMARY KEY (codVenda) ,
7     FOREIGN KEY (cpfVendedor) REFERENCES loja.vendedor (cpfVendedor) ,

```

```

8     FOREIGN KEY (cpfCli) REFERENCES loja.cliente (cpfCli)
9 );
10
11
12 CREATE TABLE loja.produto_venda (
13     codVenda INTEGER NOT NULL ,
14     codPro INTEGER NOT NULL ,
15     qtde INTEGER NOT NULL ,
16     PRIMARY KEY (codVenda, codPro),
17     FOREIGN KEY (codVenda) REFERENCES loja.venda (codVenda),
18     FOREIGN KEY (codPro) REFERENCES loja.produto (codPro)
19 );
20
21
22 CREATE TABLE loja.telefone (
23     cpfCli INTEGER NOT NULL ,
24     telefone VARCHAR(20) NOT NULL ,
25     PRIMARY KEY (cpfCli, telefone),
26     FOREIGN KEY (cpfCli) REFERENCES loja.cliente (cpfCli)
27 );

```

7.2.3 Excluindo tabelas - DROP TABLE

Para excluir a estrutura de uma tabela temos o comando `DROP TABLE` (Script ??). Este comando exclui a estrutura da tabela e, logicamente, todos os dados presentes nela. Muito cuidado com o comando `DROP TABLE`.

Script 12: Comando para exclusão de tabelas

```
1 DROP TABLE <nome_da_tabela>;
```

7.2.4 Alterações das definições de tabelas - ALTER TABLE

O comando `ALTER TABLE` permite alterar as definições de uma tabela.

Algumas ações possíveis com este comando são:

1. Renomear atributo
2. Renomear tabela
3. Adicionar atributos
4. Remover atributos
5. Alterar tipo de dado de um atributo

Veja com os exemplos a seguir algumas aplicações deste importante comando.

1. Renomear atributo

Script 13: Renomeando o atributo nome da tabela cliente para nomeCli

```
1 ALTER TABLE cliente CHANGE COLUMN nome nomeCli VARCHAR(45) NOT NULL;
```

2. Renomear tabela

Script 14: Renomeando o atributo nome da tabela cliente para nomeCli

```
1 ALTER TABLE cliente RENAME TO tab_cliente;
```

3. Adicionar atributos

Script 15: Adicionando atributos à tabela cliente

```
1 ALTER TABLE cliente ADD [COLUMN] cidade VARCHAR(45);
2 ALTER TABLE cliente ADD COLUMN (cidade VARCHAR(45), estado VARCHAR(2));
```

4. Remover atributos

Script 16: Removendo o atributo cidade da tabela cliente

```
1 ALTER TABLE cliente DROP [COLUMN] cidade;
```

5. Alterar tipo de dado de um atributo

Script 17: Modificando o tipo de dados do atributo

```
1 ALTER TABLE cliente MODIFY nome VARCHAR(100) NOT NULL;
```

Outras possibilidades com o uso do ALTER TABLE

Vamos fazer uma modificação no esquema conceitual do banco de dados da Loja. Devemos acrescentar o atributo cidade à tabela cliente, sendo que este atributo será uma chave estrangeira para a tabela cidade. Vejamos as modificações no modelo na Figura 7.3.

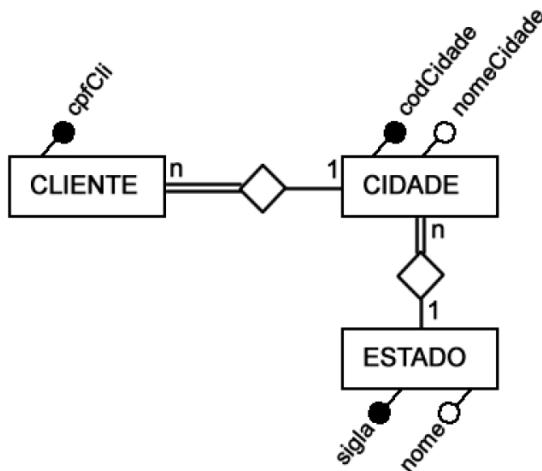


Figura 7.3: Modificação no esquema do projeto Loja.

Agora vejamos as modificações em SQL.

Script 18: Criação das tabelas cidade e estado.

```
1 CREATE TABLE estado (
2     sigla CHAR(2) NOT NULL ,
3     nome VARCHAR(50) NOT NULL ,
4     PRIMARY KEY (sigla)
```

```

5 );
6
7 CREATE TABLE cidade(
8   codCidade INTEGER UNSIGNED NOT NULL ,
9   nomeCidade VARCHAR(100) NOT NULL ,
10  sigla CHAR(2) NOT NULL ,
11  PRIMARY KEY (codCidade),
12  FOREIGN KEY fk_cidade_sigla(sigla)
13    REFERENCES estado(sigla)
14 );

```

Script 19: Alteração da tabela cliente, acrescentando a coluna codCidade e a chave estrangeira.

```

1 ALTER TABLE cliente ADD COLUMN codCidade INTEGER UNSIGNED NOT NULL ;
2
3 ALTER TABLE cliente ADD CONSTRAINT fk_cliente_codCidade
4   FOREIGN KEY(codCidade)
5     REFERENCES cidade(codCidade);

```

Outra modificação muito usada com o comando ALTER TABLE é a adição de chave primária ou chave estrangeira à tabela após sua criação. Veja no Script ??.

Script 20: Criação da chave primária e chave estrangeira após a criação da tabela.

```

1 CREATE TABLE cidade(
2   codCidade INTEGER UNSIGNED NOT NULL ,
3   nomeCidade VARCHAR(100) NOT NULL ,
4   sigla CHAR(2) NOT NULL
5 );
6
7 ALTER TABLE cidade ADD PRIMARY KEY (codCidade);
8
9 ALTER TABLE cidade ADD CONSTRAINT fk_cidade_sigla
10   FOREIGN KEY (sigla)
11     REFERENCES estado (sigla);

```

A criação de chaves após a criação das tabelas é muito usada quando temos um grande número de tabelas e restrições entre elas, implicando diretamente em saber de antemão a ordem de criação das tabelas. Se optamos por criar as restrições de integridade referencial dentro da tabela (dentro do create table), então a ordem de criação das tabelas torna-se muito importante; criamos primeiro aquelas tabelas que não referenciam outras tabelas, mas que são referenciadas por outras. No caso do exemplo do projeto Loja, uma ordem possível de criação das tabelas seria a seguinte:

1. estado
2. cidade
3. cliente
4. produto
5. vendedor

6. venda
7. produto_venda
8. telefone

Havendo, logicamente, outras ordens possíveis. O importante é que as tabelas que são referenciadas sejam criadas primeiro. Em projetos muito grandes e quando não dispomos de ferramentas CASE para construirmos o modelo físico (que geram as tabelas na ordem correta), o mais comum é criarmos todas as tabelas e suas respectivas chaves primárias e depois criarmos as restrições de chave estrangeira com o comando ALTER TABLE.

7.3 Linguagem de Manipulação de Dados (DML)

A DML - Data Manipulation Language (Linguagem de Manipulação de Dados) inclui comandos para inserir, editar, excluir ou consultar dados nas tabelas e visões. Portanto, os comandos principais que iremos estudar são: INSERT, UPDATE, DELETE e SELECT, e as suas cláusulas e variações. Para facilitar o entendimento, utilizaremos como exemplo o esquema do projeto *loja*.

7.3.1 Consultas - SELECT

Guia básico de consultas para SQGB MySQL 5.7

A seguir apresentamos um guia de consultas simplificado para o comando SELECT.

O SELECT permite recuperar dados de uma estrutura/objeto do banco de dados, como uma tabela, uma view, e em alguns casos, uma stored procedure ou função (alguns SGBD's permitem a criação de procedimentos que retornam valor). Resumidamente, o SELECT permite a seleção de tuplas e atributos em uma ou mais relações.

Os comandos apresentados abaixo referem-se à sintaxe adotada pelo MySQL na versão 5.7.

Script 21: Sintaxe básica do SELECT

```
1 SELECT <lista_de_campos>
2 FROM <nome_da_tabela>
```

Exemplo: Seja o seguinte esquema de tabela ‘cliente’

cliente(código, nome, sobrenome, dataNasc, CPF, endereço, cidade, estado, ativo)

cliente								
código	nome	sobrenome	dataNasc	CPF	endereço	cidade	estado	ativo
INT	VARCHAR	VARCHAR	DATE	INT	VARCHAR	VARCHAR	CHAR(2)	ENUM
PRIMARY KEY								

Liste todas as colunas de todos os clientes:

Script 22: Lista de todos os clientes

```
1 SELECT *
2 FROM cliente;
```

Obs: O uso de * obriga o SGBD a consultar quais são os campos antes de efetuar a busca dos dados, criando mais um passo no processo.

Liste o código e o nome de todos os clientes:

Script 23: Lista de código e nome de todos os clientes

```
1 SELECT código, nome
2 FROM cliente;
```

Liste o código e o nome completo de todos os clientes:

Script 24: Lista de código e nome completo de todos os clientes

```
1 SELECT código, CONCAT(nome, ' ', sobrenome) AS nomeCompleto
2 FROM cliente;
```

Cláusula WHERE

Permite ao comando SELECT passar condições de filtragem.

Os Script 25, 26 e 27 mostram exemplos.

Liste nome e sobrenome, de todos os clientes que são do estado do RJ.

Script 25: Consulta nome e sobrenome, de todos os clientes que são do estado do RJ

```
1 SELECT nome, sobrenome
2 FROM cliente
3 WHERE estado = 'RJ';
```

Liste nome e sobrenome, de todos os clientes que são do estado do RJ ou do estado de MG.

Script 26: Consulta nome e sobrenome, de todos os clientes que são do estado do RJ ou do estado de MG

```
1 SELECT nome, sobrenome
2 FROM cliente
3 WHERE estado = 'RJ' OR estado = 'MG';
```

Liste nome e sobrenome, de todos os clientes que são do estado do RJ ou, são do estado de MG e estão inativos.

Script 27: Consulta com condições ligadas por operador lógico

```
1 SELECT nome, sobrenome
2 FROM cliente
3 WHERE estado = 'RJ' OR (estado = 'MG' AND ativo = 'N');
```

O Script 27 retorna todos os clientes do RJ e apenas os inativos de MG.

Se precisarmos consultar os registros cujo determinado campo não foi cadastrado valor temos a opção IS NULL, no MySQL. Observe o Script 25.

Liste o cpf, nome e sobrenome, de todos os clientes cuja cidade ou data de nascimento não foram informadas.

Script 28: Lista de clientes com cidade ou data de nascimento não informadas.

```
1 SELECT cpfCli, nome, sobrenome
2 FROM cliente
3 WHERE cidade IS NULL OR dataNasc IS NULL;
```

Filtros de texto

Para busca parcial de string, o SELECT fornece o operador LIKE.

Liste nome e sobrenome, de todos os clientes cujo nome inicial é ‘Maria’.

Script 29: Lista de clientes cujo primeiro nome é ‘Maria’

```
1 SELECT nome, sobrenome
2 FROM cliente
3 WHERE nome LIKE 'Maria%';
```

Observe o uso da máscara %, que indica que pode vir qualquer outra string após ‘Maria’. Se precisamos consultar as tuplas cuja string ‘Maria’ está presente em qualquer posição do campo nome, podemos fazer ... WHERE nome LIKE '%Maria%'; E para pesquisar as tuplas cuja string ‘Maria’ está no fim do campo nome, fazemos ... WHERE nome LIKE 'Maria%';

O uso de máscara no início e no fim da string fornece maior poder de busca, mas causa considerável perda de performance. Este recurso deve ser usado com critério.

Por padrão, a SQL diferencia caixa alta de caixa baixa. Para eliminar essa diferença, utilizamos a função UPPER() ou LOWER().

Exemplo de uso das funções UPPER e LOWER

Script 30: Exemplo de uso das funções UPPER e LOWER

```

1 SELECT nome, sobrenome
2 FROM cliente
3 WHERE UPPER(nome) LIKE 'MARIA%';
4
5 SELECT nome, sobrenome
6 FROM cliente
7 WHERE LOWER(nome) LIKE 'maria%';

```

SELECT UPPER('Toni Luiz'); → TONI LUIZ

SELECT LOWER('Toni Luiz'); → toni luiz

Ordenação

A ordenação pode ser definida com a cláusula ORDER BY. Assim como na cláusula WHERE, o campo de ordenação não precisa estar listado no campo de visualização.

Seleção de todos os clientes ordenados por nome e seleção de todos os clientes ordenados por estado e por nome

Script 31: Uso do ORDER BY

```

1 SELECT cpfCli, nome
2 FROM cliente
3 ORDER BY nome;
4
5 SELECT cpfCli, nome
6 FROM cliente
7 ORDER BY estado, nome;

```

O padrão de ordenação é ASC (ou seja, ascendente, crescente). Se precisamos criar uma visualização em ordem decrescente de um campo, usamos DESC (decrescente).

Seleção de todos os clientes ordenados em ordem decrescente do sobrenome. E seleção de todos os clientes ordenados em ordem alfabética de cidade e decrescente de nome.

Script 32: Listas ordenadas de modo crescente e decrescente

```

1 SELECT cpfCli, nome, sobrenome
2 FROM cliente
3 ORDER BY sobrenome DESC;
4
5 SELECT cpfCli, nome, cidade
6 FROM cliente
7 ORDER BY cidade ASC, nome DESC;

```

Exemplo 1: Selecionar o nome e o NSS dos empregados que trabalham no departamento 2.

```
SELECT Pnome, Mnome, Snome, NSS
FROM Empregado
WHERE NDEP=2;
```

Esta operação corresponde ao seguinte cálculo em álgebra relacional:

$$\pi_{Pnome, Mnome, Snome, NSS}(\sigma_{NDEP=2}(EMPREGADO))$$

Exemplo 2: Selecionar o nome e o NSS dos empregados que trabalham no departamento 2 e com salário > 2500,00

```
SELECT Pnome, Mnome, Snome, NSS
FROM Empregado
WHERE NDEP=2
    AND salario > 2500;
```

SELECT com junção de tabelas

O SELECT permite juntar 2 ou mais tabelas no mesmo resultado. Isso pode ser feito de várias formas.

Para exemplificar, vamos utilizar o seguinte esquema do diagrama entidade relacionamento:

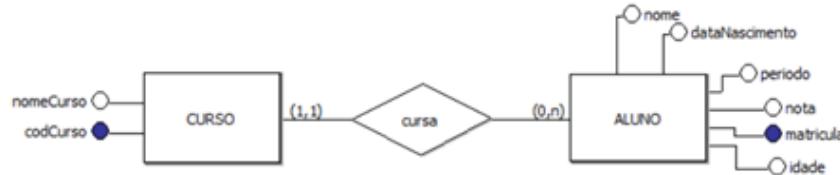


Figura 7.4: Modelo entidade relacionamento aluno-curso

Isso resulta nos esquemas de tabela abaixo:

```
curso (codCurso, nomeCurso)
aluno (matricula, nome, dataNascimento, período, idade, codCurso - FK curso)
# Liste matrícula, nome do aluno e nome do curso, de todos os alunos. Ordene pelo nome do curso e nome do aluno
```

Script 33: Exemplo de junção com operação de igualdade

```
1 SELECT matricula, nome, nomeCurso
2 FROM aluno a, curso c
3 WHERE a.codCurso = c.codCurso
4 ORDER BY nomeCurso, nome;
```

As tabelas aluno e curso são unificadas através do campo chave (chave estrangeira), em uma operação de igualdade.

Cláusula JOIN

A junção de tabelas no comando SELECT também pode ser feita com a cláusula JOIN. Esse comando deve ser usado com a palavra reservada INNER ou OUTER.

INNER: Semelhante ao uso do operador "=" na junção de tabelas. Aqui os registros sem correspondências não são incluídos.

OUTER: Os registros que não se relacionam também são exibidos. Neste caso, é possível definir qual tabela (à esquerda ou à direita – LEFT ou RIGHT) será incluída na seleção, mesmo não tendo correspondência.

Veja que no diagrama de exemplo (aluno – curso) podemos ter um curso que ainda não tem alunos (relacionamento parcial do lado de curso). Sendo assim, deseja-se obter um relatório de todos os cursos e seus respectivos alunos, se houver.

Se existir um curso que não tenha aluno matriculado, este curso deve ser exibido.

Suponha para este caso a seguinte situação:

curso	
codCurso	nomeCurso
1	Redes de Computadores
2	Eletrotécnica
3	Mecatrônica
4	Engenharia Elétrica

aluno				
matricula	nome	codCurso	periodo	nota
1	Fulano	1	1	6
2	Beltrano	2	2	7
3	Ciclano	1	3	5
4	Toni Luiz	3	1	7

Observe que o curso de Engenharia Elétrica não tem alunos cadastrados.

Se fizermos a consulta usando INNER JOIN vamos listar somente os cursos que tem alunos, que não é exatamente o que queremos neste exemplo. Veja no Script 34.

Script 34: Consulta de alunos e cursos usando INNER JOIN

```

1 SELECT matricula, nome, a.codCurso, nomeCurso
2 FROM aluno a
3 INNER JOIN curso c
4   ON a.codCurso = c.codCurso;

```

* Observe que a condição de junção é feita na cláusula ON do JOIN.

A consulta do Script 34 vai retornar o seguinte resultado para o nosso exemplo:

matricula	nome	codCurso	nomeCurso
1	Fulano	1	Redes de Computadores
2	Beltrano	2	Eletrotécnica
3	Ciclano	1	Redes de Computadores
4	Toni Luiz	3	Mecatrônica

Não retornou nenhum registro (mesmo que com os outros campos vazios) para o curso de Engenharia Elétrica, porque não houve correspondência com aluno.

Para resolver o problema usaremos OUTER JOIN (ao pé da letra significa “junção externa”)

Script 35: Consulta de alunos e cursos usando OUTER JOIN

```

1 SELECT matricula, nome, c.codCurso, nomeCurso
2 FROM aluno a
3 RIGHT OUTER JOIN curso c
4 ON a.codCurso = c.codCurso;

```

* RIGHT porque a tabela prioritária para nós (curso) está à direita no JOIN.

Agora, essa consulta resulta em:

matricula	nome	codCurso	nomeCurso
1	Fulano	1	Redes de Computadores
2	Beltrano	2	Eletrotécnica
3	Ciclano	1	Redes de Computadores
4	Toni Luiz	3	Mecatrônica
		4	Engenharia Elétrica

Exemplo 3: Selecionar o nome e o NSS dos empregados que são supervisores.

```

SELECT e1.Pnome, e1.Mnome, e1.Snome, e1.NSS
FROM Empregado e1, Empregado e2
WHERE e1.NSS=e2.NSSSuper;

```

e1 e e2 são chamados “aliases” e representam a relação a qual estão referenciando.

Esta operação corresponde ao seguinte cálculo em álgebra relacional:

$\pi_{Pnome, Mnome, Snome, NSS}(EMPREGADO \bowtie_{NSS=NSSSuper} EMPREGADO)$

Outros exemplos:

Seleciona todos os atributos (e tuplas) da relação Empregado:

```

SELECT *
FROM Empregado;

```

Seleciona o atributo NDEP, de todas as tuplas da relação Empregado:

```

SELECT NDEP
FROM Empregado;

```

Seleciona o atributo NDEP, de todas as tuplas da relação Empregado, sem repetição:

```

SELECT DISTINCT NDEP
FROM Empregado;

```

Seleciona os atributos NDEP e Pnome, de todas as tuplas da relação Empregado, sem repetição:

```

SELECT DISTINCT NDEP, Pnome
FROM Empregado;

```

	Pnome	...	NDEP
1	J		1
2	A		3
3	P		4
4	M		5
5	M		4
6	J		1
7	B		2

Observe as linhas 1 e 6. Como os atributos Pnome e NDEP, juntos, são repetidos, então a linha 6 não será retornada na consulta.

Selecione o nome dos Empregados que trabalham em projetos localizados em “Houston”.

```
SELECT Pnome, Mnome, Snome
FROM Empregado
WHERE NSS IN (SELECT NSSEMP
               FROM Trabalha_Em
               WHERE PNRO IN (SELECT Pnumero
                               FROM Projeto
                               WHERE Localizacao='Houston'
                           )
             );

```

$$R_1 \leftarrow \pi_{Pnumero}(\sigma_{\text{Localização}=\text{"Houston"}(\text{Projeto})})$$

Em álgebra relacional, temos: $R_2 \leftarrow \pi_{NSSEMP}(Trabalha_Em \bowtie_{PNRO=Pnumero} R_1)$

$$RESULTADO \leftarrow \pi_{Pnome, Mnome, Snome}(Empregado \bowtie_{NSS=NSSEMP} R_2)$$

Selecione os empregados em ordem alfabética:

```
SELECT *
FROM Empregado
ORDER BY Pnome;
```

FULL OUTER JOIN

A SQL padrão contempla a cláusula FULL OUTER JOIN, a qual retorna à esquerda e à direita dos conjuntos de dados envolvidos na consulta. No entanto, esta opção não é contemplada no MySQL, para a versão utilizada nesta apostila. Para simular o comportamento de FULL OUTER JOIN, no MySQL, usamos como recurso a cláusula UNION (união). Por exemplo:

```
SELECT *
FROM tabela_a
LEFT OUTER JOIN tabela_b
  ON tabela_a.id = tabela_b.id
```

UNION

```
SELECT *
FROM tabela_a
RIGHT OUTER JOIN tabela_b
  ON tabela_a.id = tabela_b.id;
```

Neste caso, realizamos JOINS para ambas as direções: LEFT e RIGHT

Unindo os JOINS (UNION) ganhamos a mesma funcionalidade de um FULL OUTER JOIN, que é o que estamos precisando

Agrupamentos

Um poderoso recurso do comando SELECT é a declaração GROUP BY. É frequentemente usado com funções de agregação (COUNT, MAX, MIN, SUM, AVG) para agrupar os conjuntos de resultados por uma ou mais colunas.

Liste o nome do curso e sua quantidade de alunos. Ordene pela quantidade de alunos, decrescente.

Script 36: Consulta da quantidade de alunos por curso, usando INNER JOIN

```
1 SELECT nomeCurso, COUNT(a.codCurso) as numAlunos
2 FROM curso c
3     INNER JOIN aluno a ON c.codCurso = a.codCurso
4 GROUP BY nomeCurso
5 ORDER BY COUNT(*) DESC;
```

* Usando INNER JOIN não iremos listar os cursos que não tem alunos ...

Para corrigir o problema usamos OUTER JOIN (Script 37).

Script 37: Consulta da quantidade de alunos por curso, usando OUTER JOIN

```
1 SELECT nomeCurso, COALESCE(COUNT(a.codCurso),0) as numAlunos
2 FROM curso c
3     LEFT OUTER JOIN aluno a ON c.codCurso = a.codCurso
4 GROUP BY nomeCurso
5 ORDER BY COUNT(*) DESC;
```

* A função coalesce é utilizada para, se retornar NULL na contagem, coloque 0 no lugar.

Funções de Agregação

COUNT: retorna a quantidade de itens da seleção

AVG: retorna a média do campo especificado

MIN, MAX e SUM: retornam, respectivamente, o menor, o maior e o somatório de um grupo de registros.

Arredondamentos

Para fazer arredondamentos de valores numéricos usamos, geralmente, a função ROUND. A função ROUND recebe o nome do campo e o número de casas decimais para o arredondamento.

Exemplo: O Script 38 retorna a menor nota, a maior nota, a média das notas e o somatório, agrupados por curso.

Script 38: Consulta com funções de agregação e agrupamento

```
1 SELECT nomeCurso, MIN(nota) as 'menor nota',
2       MAX(nota) as 'maior nota',
3       AVG(nota) as 'media',
4       SUM(nota) as 'soma'
5 FROM aluno a, curso c
```

```

6 WHERE a.codCurso = c.codCurso
7 GROUP BY nomeCurso;

```

Num determinado momento, a consulta retorna os seguintes valores, mostrados na Figura 7.5.

nomeCurso	menor nota	maior nota	média	soma
Eletrotécnica	6.00	8.50	7.400000	22.20
Engenharia Elétrica	4.70	10.00	7.500000	22.50
Mecatrônica	1.25	8.00	3.616667	10.85
Redes de Computadores	2.20	7.73	4.291667	25.75

Figura 7.5: Resultado do Script 38

Vamos melhorar a saída, aplicando um arredondamento de 2 casas decimais na média, Script 39

Script 39: Consulta com arredondamento

```

1 SELECT nomeCurso,
2     MIN(nota) as 'menor nota',
3     MAX(nota) as 'maior nota',
4     ROUND(AVG(nota),2) as 'media',
5     SUM(nota) as 'soma',
6 FROM aluno a, curso c
7 WHERE a.codCurso = c.codCurso
8 GROUP BY nomeCurso;

```

Agora suponha que queremos exibir somente os cursos cuja média é superior a 6. Para isso, precisamos fazer um filtro após o agrupamento. Neste caso, temos a cláusula HAVING.

HAVING

O HAVING foi adicionado a SQL porque a cláusula WHERE não pode ser usada com funções de agrupamento.

Portanto, se WHERE é o filtro do FROM, o HAVING é o filtro do GROUP BY.

Então, para resolver nosso problema de exibir somente os cursos cuja média é superior a 6, fazemos como no Script 40.

Script 40: Consulta com agrupamento filtrado com o HAVING

```

1 SELECT nomeCurso,
2     MIN(nota) as 'menor nota',
3     MAX(nota) as 'maior nota',
4     ROUND(AVG(nota),2) as 'media',
5     SUM(nota) as 'soma',
6 FROM aluno a, curso c
7 WHERE a.codCurso = c.codCurso
8 GROUP BY nomeCurso
9 HAVING AVG(nota) > 6;

```

nomeCurso	menor nota	maior nota	média	soma
Eletrotécnica	6.00	8.50	7.40	22.20
Engenharia Elétrica	4.70	10.00	7.50	22.50

Figura 7.6: Resultado do Script 40

Num determinado momento, o Script 40 tem o resultado mostrado na Figura 7.6.
Outras funções matemáticas que podem ser úteis

CEIL: retorna o menor inteiro que é maior ou igual a um número (retorna o teto)

Exemplo:

SELECT CEIL(25.75); → Resultado: 26

FLOOR: retorna o maior inteiro que é menor ou igual a um número (retorna o piso)

Exemplo:

SELECT FLOOR(25.75); → Resultado: 25

ABS: retorna o valor absoluto (sem sinal)

Exemplo:

SELECT ABS(-25.75); → Resultado: 25.75

7.3.2 Inserções - INSERT

Inserção de um novo dado (nova tupla) numa tabela do banco de dados.

Sintaxe básica (Script):

Script 41: Sintaxe básica de INSERT

```
1 INSERT INTO <RELACAO> (<Atributos>)
2 VALUES (<valores>);
```

Exemplos:

- Quando coloca valores para todos os atributos.

Script 42: INSERT com valores para todos os atributos da tabela

```
1 INSERT INTO EMPREGADO
2 VALUES ('Cecilia', 'F', ..., 4);
```

* Observação: Neste caso, a ordem dos valores deve obedecer a ordem de criação dos atributos na tabela.

- Quando especifica os atributos que receberão valores:

Script 43: INSERT com valores de atributos específicos

```
1 INSERT INTO EMPREGADO (Pnome, Mnome, Snome, Sexo, Idade, ...)
2 VALUES ('Cecilia', '', 'Santos', 'F', 22, ...);
```

* Observação: Atributos criados com a opção NOT NULL, devem ter sempre seus valores especificados, já que não podem ser nulos.

7.3.3 Atualizações - UPDATE

Edição de dados já existentes numa tabela.

Sintaxe básica:

Script 44: Sintaxe básica do UPDATE

```
1 UPDATE <RELACAO>
2 SET <Atributo> = <Expressao>
3 WHERE <Condicao>;
```

Exemplo:

Atualiza o salário de todos os empregados do departamento 2 para 3000,00.

Script 45: INSERT com valores de atributos específicos

```
1 UPDATE EMPREGADO
2 SET Salario = 3000
3 WHERE NDEP = 2;
```

* No caso de mais de um atributo que terá valores editados, o par atributo = valor são separados por vírgula.

7.3.4 Remoção/exclusão de dados - DELETE

Remover tuplas na tabela. Comando DELETE.

Sintaxe básica:

Script 46: Sintaxe básica do DELETE

```
1 DELETE FROM <RELACAO>
2 WHERE <Condicao>;
```

Exemplo:

Script 47: Exemplo de DELETE na tabela Empregado, com condições

```
1 DELETE FROM EMPREGADO
2 WHERE Salario < 2500 AND NDEP = 3;
```

7.3.5 Restrições sobre as cláusulas DELETE e UPDATE

No momento da definição de uma tabela ou até mesmo pós definição, podemos impor restrições sobre as cláusulas UPDATE e/ou DELETE. Essas restrições indicam ao SGBD como deve se comportar na execução de uma dessas cláusulas.

Vimos que, a integridade referencial é especificada por meio da cláusula FOREIGN KEY (chave estrangeira). Além disso, vimos também que uma restrição de integridade referencial pode ser violada quando tuplas são inseridas ou excluídas, ou quando um valor de atributo de chave estrangeira ou chave primária é modificado. A ação *default* que a SQL (no nosso caso, o MySQL) toma para uma violação de integridade é **rejeitar** a operação de atualização ou exclusão que causará uma violação, o que é conhecido como opção RESTRICT.

Restrições sobre a cláusula DELETE, suportadas pelo MySQL

- ON DELETE RESTRICT (default)

- ON DELETE CASCADE
- ON DELETE SET NULL

Restrições sobre a cláusula UPDATE, suportadas pelo MySQL

- ON UPDATE RESTRICT (default)
- ON UPDATE CASCADE

Exemplos:

```
FOREIGN KEY (NSSSUPER) REFERENCES
    EMPREGADO (NSS)
    ON DELETE SET NULL ON UPDATE CASCADE;
```

Neste caso, ocorrendo exclusão na tabela EMPREGADO, as chaves estrangeiras que referenciam a chave NSS, terão seus valores setados com NULL (opção ON DELETE SET NULL). No caso de atualização na tabela EMPREGADO, o novo valor atribuído à chave NSS será replicado a todos os campos que se fazem referência a ela (opção ON UPDATE CASCADE).

```
FOREIGN KEY (NDEP) REFERENCES
    DEPARTAMENTO (DNUMERO)
    ON DELETE CASCADE ON UPDATE CASCADE;
```

Neste caso, ocorrendo exclusão na tabela EMPREGADO, as chaves estrangeiras que referenciam a chave NSS, também serão excluídas (opção ON DELETE CASCADE). No caso de atualização na tabela EMPREGADO, o novo valor atribuído à chave NSS será replicado a todos os campos que se fazem referência a ela (opção ON UPDATE CASCADE).

Portanto, Na opção CASCADE temos a replicação da ação ou do valor. E na opção SET NULL, será atribuído o valor NULL aos campos que referenciam a chave, na ocorrência de exclusão da chave referenciada.

7.3.6 Outras cláusulas da SQL

Apresentamos nesta seção algumas cláusulas de consulta em SQL, que podem ser úteis. Observações:

1. CONTAINS

CONTAINS compara dois conjuntos de valores e retorna TRUE se um conjunto contém todos os valores do outro. Exemplo: Encontrar o nome completo de empregados que trabalham em todos os projetos controlados pelo departamento 5. (Em álgebra relacional é característica de divisão.)

```
SELECT Pnome, Mnome, Snome
FROM Empregado
WHERE ((SELECT PNRO
        FROM Trabalha_Em
        WHERE NSS=NSSEMP)
        CONTAINS
        (SELECT Pnumero
        FROM Projeto
```

```

        WHERE DNUM = 5
    )
);

```

A segunda consulta dentro do parênteses recupera todos os números de projetos controlados pelo departamento 5. Para cada tupla de empregado, a primeira consulta dentro do parênteses recupera os números de projetos sobre os quais os empregados trabalham. Se o resultado desta consulta contiver todos os projetos controlados pelo departamento 5, a tupla de empregado é selecionada e o nome deste empregado é apresentado.

Note que CONTAINS é similar a operação divisão da álgebra relacional.

2. EXISTS (NOT EXISTS)

EXISTS (NOT EXISTS) é utilizado para verificar se o resultado de uma subconsulta correlacionada é ou não vazia. EXISTS(Q) declara TRUE se existe pelo menos uma tupla no resultado da consulta Q e declara FALSE em caso contrário. NOT EXISTS(Q) declara TRUE se não existir tupla no resultado da consulta Q e declara FALSE em caso contrário. Exemplo: Listar o nome completo dos empregados que não possuem dependentes.

```

SELECT Pnome, Mnome, Snome
FROM Empregado
WHERE NOT EXISTS (SELECT *
                   FROM Dependente
                   WHERE NSS=NSSEMP);

```

7.4 Exercícios

- O comando:

```

CREATE TABLE livro (
    issn integer not null primary key,
    titulo varchar(100) not null,
    numPaginas tinyint(4)
);

```

Escolha uma:

- (a) Há um erro na sintaxe.
 - (b) Cria o esquema de banco de dados livro.
 - (c) Altera a tabela livro adicionando as colunas issn, titulo e numPaginas.
 - (d) Cria a tabela livro, com três colunas, sendo a chave primária issn.
- Para criação de um banco de dados chamado biblioteca, em SQL temos o comando:
Escolha uma ou mais:
 - (a) CREATE USER biblioteca;
 - (b) CREATE DATABASE biblioteca;

- (c) CREATE SCHEMA biblioteca;
- (d) CREATE TABLE biblioteca();

3. Suponha que precisamos criar as relações abaixo no banco de dados biblioteca.

```
livro(issn, titulo, subtítulo, codAutor, numPáginas, codEditora)
editora(codEditora, nomeEditora)
autor(codAutor, nomeAutor, sobrenomeAutor)
```

Para criar todas essas estruturas, qual a ordem necessária para os comandos?

Escolha uma:

- (a) Os comandos executados em qualquer ordem produzirão o mesmo resultado.
- (b) CREATE SCHEMA biblioteca; CREATE TABLE livro; CREATE TABLE editora; CREATE TABLE autor;
- (c) CREATE TABLE autor; CREATE TABLE editora; CREATE TABLE livro; CREATE SCHEMA biblioteca;
- (d) CREATE SCHEMA biblioteca; CREATE TABLE autor; CREATE TABLE editora; CREATE TABLE livro;

4. Escreva os comandos em SQL para criação do banco de dados biblioteca e das relações listadas abaixo:

```
livro(issn, titulo, subtítulo, codAutor, numPáginas, codEditora)
editora(codEditora, nomeEditora)
autor(codAutor, nomeAutor, sobrenomeAutor)
```

5. Dado o esquema:

```
livro(issn, titulo, subtítulo, codAutor, numPáginas, codEditora)
editora(codEditora, nomeEditora)
autor(codAutor, nomeAutor, sobrenomeAutor)
```

Sobre a relação livro e autor, podemos afirmar que (Escolha uma ou mais):

- (a) Não é possível estabelecer a razão de cardinalidade somente com as informações do esquema relacional
- (b) Um autor pode escrever vários livros.
- (c) Um autor pode escrever vários livros e um livro pode ter vários autores.
- (d) Um autor pode escrever somente um livro
- (e) Um livro pode ter somente um autor

6. Dado o esquema do banco de dados biblioteca:

```
livro(issn, titulo, subtítulo, codAutor, numPáginas, codEditora)
editora(codEditora, nomeEditora)
autor(codAutor, nomeAutor, sobrenomeAutor)
```

Escreva os comandos SQL necessários para permitir que o relacionamento entre autor e livro seja n:n. Ou seja, defina as relações necessárias para permitir que um livro possa ter mais de um autor e um autor possa escrever mais de um livro.

7. Seja a tabela autor, definida pelo SQL:

```
CREATE TABLE autor(
    codAutor integer not null primary key,
    nomeAutor varchar(50) not null,
    sobrenomeAutor varchar(50) not null
);
```

Qual o comando necessário para alterar a tabela e adicionar a coluna apelidoAutor?

Escolha uma ou mais:

- (a) ALTER TABLE apelidoAutor ADD COLUMN autor;
 - (b) ALTER TABLE autor ADD COLUMN apelidoAutor NULL;
 - (c) ALTER TABLE autor ADD COLUMN apelidoAutor;
 - (d) ALTER TABLE autor ADD apelidoAutor VARCHAR(50);
 - (e) ALTER TABLE autor ADD COLUMN apelidoAutor VARCHAR(50);
8. Para o esquema de banco de dados biblioteca mostrado abaixo, é necessário também manter a informação do assunto principal de cada livro.

```
livro(isbn, titulo, subtítulo, numPaginas, codEditora);
editora(codEditora, nomeEditora);
autor(codAutor, nomeAutor, sobrenomeAutor);
autor_livro(isbn, codAutor);
```

Escreva os scripts SQL para definir as estruturas necessárias para alterar este esquema e armazenar a informação do assunto principal de cada livro. Considere uma tabela para cadastrar os assuntos principais.

9. Para o esquema biblioteca, abaixo:

```
livro(isbn, titulo, subtítulo, numPaginas, codEditora);
editora(codEditora, nomeEditora);
autor(codAutor, nomeAutor, sobrenomeAutor);
autor_livro(isbn, codAutor);
```

Escreva o comando SQL para incluir a coluna ‘ano’ na tabela livro.

10. Usando o schema biblioteca do Laboratório de Banco de Dados, faça as consultas SQL para o que se pede em cada item:
 - (a) Liste isbn, titulo e subtítulo de todos os livros, ordenados em ordem alfabética do título.
 - (b) Liste isbn, titulo e nome da Editora de cada livro, ordenado pelo nome da editora e título do livro

- (c) Liste a quantidade de livros para cada editora. Considere somente aquelas editoras que possuem algum livro cadastrado. Ordene pelo quantidade, em ordem decrescente.
- (d) Liste o nome de todas as editoras que têm livros cadastrados
- (e) Liste o nome das editoras que não têm livros cadastrados.
- (f) Liste a quantidade de livros para cada editora. Considere todas as editoras cadastradas. Caso alguma editora ainda não possua um livro editado, considere a quantidade igual a 0 (zero). Ordene pelo nome da editora.
- (g) Liste o nome de todos os assuntos que possuem algum livro cadastrado, e a quantidade de livros para cada assunto
- (h) Liste o issn e titulo de todos os livros escritos por mais de um autor
- (i) Para cada autor, liste seu nome completo (nome+sobrenome) e a quantidade de livros cadastrados. Se não houver livro de um determinado autor, retorne 0.
- (j) Liste todos os autores cuja quantidade de livros é maior ou igual que 2.

8 Dependência Funcional e Normalização

Cada esquema de relação consiste em uma série de atributos, e o esquema de banco de dados relacional consiste em uma série de esquemas de relação. Até aqui, assumimos que os atributos são agrupados para formar m esquema de relação usando o bom senso do projetista de banco de dados ou mapeando um projeto de esquema de banco de dados com base no modelo de dados conceitual, como o modelo ER ou ER Estendido (EER). Esses modelos fazem o projetista identificar os tipos de entidade e de relacionamento e seus respectivos atributos, o que leva a um agrupamento natural e lógico dos atributos em relações quando os procedimentos de mapeamento são seguidos. Porém, ainda precisamos de algum modo formal de análise porque um agrupamento de atributos em um esquema de relação pode ser melhor do que outro.

Discutiremos parte da teoria que foi desenvolvida com o objetivo de avaliar esquemas relacionais para a qualidade do projeto - ou seja, para medir formalmente por que um conjunto de agrupamentos de atributos em esquemas de relação é melhor do que outro.

Existem dois níveis em que podemos discutir as boas práticas de esquemas de relação. O primeiro é o nível lógico (ou conceitual) - como os usuários interpretam os esquemas de relação e o significado de seus atributos. Ter bons esquemas de relação nesse nível permite que os usuários entendam claramente o significado dos dados nas relações, e daí formulem suas consultas corretamente. O segundo é o nível de implementação (ou armazenamento físico) - como as tuplas em uma relação da base são armazenadas e atualizadas.

O projeto de banco de dados relacional por fim produz um conjunto de relações. Os objetivos implícitos da atividade de projeto são preservação da informação e redundância mínima. A informação é muito difícil de se quantificar - logo, consideramos a preservação de informação em matéria de manutenção de todos os conceitos, incluindo tipos de atributo, tipos de entidade e tipos de relacionamento. Assim, o projeto relacional precisa preservar todos esses conceitos, que são capturados originalmente no projeto conceitual para lógico. A minimização da redundância implica diminuir o armazenamento redundante da mesma informação e reduzir a necessidade de múltiplas atualizações para manter a consistência entre diversas cópias da mesma informação, em resposta a eventos do mundo real que exijam fazer uma atualização.

8.1 Diretrizes de projeto informais para esquemas de relação

Diretrizes informais que podem ser usadas como medidas para determinar a qualidade de projeto do esquema de relação:

- Garantir que a semântica dos atributos seja clara no esquema.
- Reduzir a informação redundante nas tuplas.
- Reduzir os valores NULL nas tuplas.
- Reprová-la possibilidade de gerar tuplas falsas.

8.1.1 Semântica clara aos atributos nas relações

Sempre que agrupamos atributos para formar um esquema de relação, consideramos que aqueles atributos pertencentes a uma relação têm certo significado no mundo real e uma interpretação apropriada associada a eles. A semântica de uma relação refere-se a seu significado resultante da interpretação dos valores de atributo em uma tupla. Uma relação pode ser interpretada como um

conjunto de fatos. Se o projeto conceitual for feito cuidadosamente e o procedimento de mapeamento for seguido de maneira sistemática, o projeto do esquema relacional deverá ter um significado claro.

Em geral, quanto mais fácil for de explicar a semântica da relação, melhor será o projeto do esquema de relação.

Exemplo, considere a Figura 8.1, uma versão simplificada do esquema de banco de dados relacional EMPRESA e a Figura 8.2, apresenta um exemplo de estados de relação preenchidos desse esquema.



Figura 8.1: Esquema de banco de dados relacional 'Empresa' simplificado

Todos os esquemas de relação na Figura 8.1 podem ser considerados fáceis de explicar e, portanto, bons do ponto de vista de ter uma semântica clara. Assim, podemos formular as seguintes diretrizes de projeto informal:

Assim, podemos formular a seguinte diretriz de projeto informal:

Diretriz 1: Projete uma tabela de modo que seja fácil explicar seu significado. Não combine atributos de vários tipos de entidade e de relacionamento em uma única tabela. Um exemplo de violação dessa diretriz pode ser visto nas Figuras 8.3 e 8.4.

8.1.2 Informação redundante nas tuplas e anomalias de atualização

Um objetivo do projeto de esquemas é minimizar o espaço de armazenamento usado pelas tabelas (e, portanto, pelos arquivos correspondentes).

O armazenamento de junções naturais de relações da base leva a um problema adicional conhecido como anomalias de atualização. Estas podem ser classificadas em anomalias de inserção, anomalias de exclusão e anomalias de modificação.

Diretriz 2: Projete os esquemas de tabelas de modo que nenhuma anomalia de inserção, exclusão ou modificação esteja presente nas tabelas. Se houver alguma anomalia⁷, anote-as claramente

⁷ Algumas considerações dos programas de aplicação podem determinar e tornar certas anomalias inevitáveis.

FUNCIONARIO					
Fnome	Cpf	Datanasc	Endereco	Dnumero	
Silva, Jose B.	12345678906	09-01-1965	Rua das Flores, 751, São Paulo, SP	5	
Wong, Fernando T.	33344555587	08-12-1955	Rua da Lapa, 34, São Paulo, SP	5	
Zelya, Alice J.	99988777767	19-01-1968	Rua Souza Lima, 35, Curitiba, PR	4	
Souza, Jennifer S.	98765432168	20-06-1941	Av. Arthur de Lima, 54, Santo André, SP	4	
Lima, Ronaldo K.	66688994476	15-09-1962	Rua Rebouças, 65, Praia Grande, SP	5	
Leite, Joice A.	45345345376	31-07-1972	Av. Lucas Otávio, 74, São Paulo, SP	5	
Pereira, André V.	98798798733	29-03-1969	Rua Timbiras, 35, São Paulo, SP	4	
Brito, Jorge E.	88866555576	10-11-1937	Rua do Horto, 35, São Paulo, SP	1	

DEPARTAMENTO		
Dnome	Dnumero	Cpf_gerente
Pesquisa	5	33344555587
Administração	4	98765432168
Matriz	1	88866555576

LOCALIZACAO_DEP	
Dnumero	Dlocal
1	São Paulo
4	Mauá
5	Santo André
5	Ilu
5	São Paulo

TRABALHA_EM		
Cpf	Projnumero	Horas
12345678906	1	32,5
12345678906	2	7,5
66688994476	3	40,0
45345345376	1	20,0
45345345376	2	20,0
33344555587	2	10,0
33344555587	3	10,0
33344555587	10	10,0
33344555587	20	10,0
99988777767	30	30,0
99988777767	10	10,0
98798798733	10	35,0
98798798733	30	5,0
98765432168	30	20,0
98765432168	20	15,0
88866555576	20	NULL

PROJETO			
Projnome	Projnumero	Projlocal	Dnum
ProdutoX	1	Santo André	5
ProdutoY	2	Ilu	5
ProdutoZ	3	São Paulo	5
Informatização	10	Mauá	4
Reorganização	20	São Paulo	1
Novosbenefícios	30	Mauá	4

Figura 8.2: Exemplo de estado de banco de dados para o esquema relacional da Figura 8.1

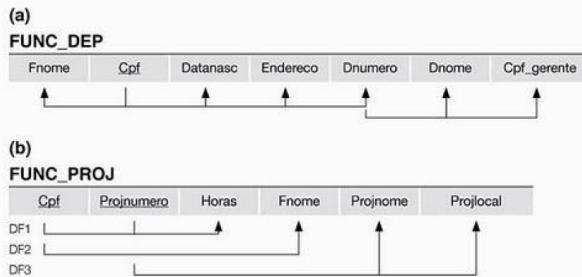


Figura 8.3: Exemplos de esquemas de tabelas sofrendo anomalias de atualização

e cuide para que os programas que atualizam o banco de dados operem corretamente.

8.1.3 Valores NULL nas tuplas

Em alguns projetos de esquema podemos agrupar muitos atributos em uma relação 'gorda'. Se muitos dos atributos não se aplicarem a todas as tuplas na relação, acabamos com muitos NULLs nessas tuplas. Isso pode desperdiçar espaço no nível de armazenamento e também ocasionar problemas com o conhecimento do significado dos atributos e com a especificação de operações 'junção' no nível lógico. Outro problema com NULLs é como considerá-los quando operações de agregação como conta ou soma são aplicadas. Operações de seleção e junção envolvem comparações; se valores NULL estiverem presentes, os resultados podem se tornar imprevisíveis. Além do mais, os NULLs podem ter várias interpretações.

Diretriz 3: Ao máximo possível, evite colocar atributos em uma tabela cujos valores podem ser NULL com frequência. Se os NULLs forem inevitáveis, garanta que eles se apliquem apenas em casos excepcionais, e não à maioria das tuplas na tabela. Usar o espaço de modo

FUNC_DEP						Redundância
Fnome	Cpf	Datanae	Endereco	Drumere	Dname	Cpf_gerente
Silva, João B.	12345678906	09-01-1965	Rua das Flores, 751, São Paulo, SP	5	Pesquisa	33344555587
Wong, Fernando T.	33344555587	08-12-1955	Rua da Lapa, 34, São Paulo, SP	5	Pesquisa	33344555587
Zelaya, Alice J.	99988777767	19-01-1968	Rua Souza Lima, 35, Cunha, PR	4	Administração	98765432168
Souza, Jennifer S.	98765432168	20-06-1941	Av. Arthur de Lima, 54, Santo André, SP	4	Administração	98765432168
Lima, Ronaldo K.	66668444476	15-09-1962	Rua Rebouças, 65, Piracicaba, SP	5	Pesquisa	33344555587
Leite, Joice A.	45345345376	31-07-1972	Av. Lucas Obes, 74, São Paulo, SP	5	Pesquisa	33344555587
Pereira, André V.	98798798733	29-03-1969	Rua Timbiras, 35, São Paulo, SP	4	Administração	98765432168
Brito, Jorge E.	88866555576	10-11-1937	Rua do Horto, 35, São Paulo, SP	1	Matriz	88866555576

FUNC_PROJ						Redundância	Redundância
Cpf	Projnumero	Horas	Fnome	Projnome	Projlocal		
12345678906	1	32.5	Silva, João B.	ProdutoX	Santo André		
12345678906	2	7.5	Silva, João B.	ProdutoY	Itu		
66668444476	3	40.0	Lima, Ronaldo K.	ProdutoZ	São Paulo		
45345345376	1	20.0	Leite, Joice A.	ProdutoX	Santo André		
45345345376	2	20.0	Leite, Joice A.	ProdutoY	Itu		
33344555587	2	10.0	Wong, Fernando T.	ProdutoY	Itu		
33344555587	3	10.0	Wong, Fernando T.	ProdutoZ	São Paulo		
33344555587	10	10.0	Wong, Fernando T.	Informatização	Mauá		
33344555587	20	10.0	Wong, Fernando T.	Reorganização	São Paulo		
99988777767	30	30.0	Zelaya, Alice J.	Novosbenefícios	Mauá		
99988777767	10	10.0	Zelaya, Alice J.	Informatização	Mauá		
98798798733	10	35.0	Pereira, André V.	Informatização	Mauá		
98798798733	30	5.0	Pereira, André V.	Novosbenefícios	Mauá		
98765432168	30	20.0	Souza, Jennifer S.	Novosbenefícios	Mauá		
98765432168	20	15.0	Souza, Jennifer S.	Reorganização	São Paulo		
88866555576	20	Nula	Brito, Jorge E.	Reorganização	São Paulo		

Figura 8.4: Exemplos de estados para as tabelas da Figura 8.3

eficaz e evitar junções com valores NULL são os dois critérios prioritários que determinam a inclusão das colunas que podem ter NULLs em uma relação ou que podem ter uma relação separada para essas colunas (com as colunas de chave apropriadas). Por exemplo, se apenas 15 por cento dos funcionários têm escritórios individuais, há pouca justificativa para incluir um atributo *Numero_escritorio* na relação FUNCIONÁRIO. Em vez disso, uma relação FUNC_ESCRITORIO (*Fcpf*, *Numero_escritorio*) pode ser criada para incluir tuplas apenas para funcionários com escritórios individuais.

8.1.4 Geração de tuplas falsas

Considere os esquemas de duas relações FUNC_LOCAL e FUNC_PROJ1 da Figura 8.5a, que podem ser usados no lugar da única relação FUNC_PROJ da Figura 8.3b. Uma tupla em FUNC_LOCAL significa que o funcionário cujo nome é *Fnome* trabalha em *algum* projeto cujo local é *Projlocal*. Uma tupla em *FUNC_PROJ1* refere-se ao fato de o funcionário cujo número de Cadastro de Pessoa Física é *Cpf* trabalhar *Horas* por semana noa projeto cujo nome, número e localização são *Projnome*, *Projnumero* e *Projlocal*. A Figura 8.5b mostra os estados da reação de FUNC_LOCAL e FUNC_PROJ1 correspondentes à relação FUNC_PROJ da Figura 8.4, que são obtidos aplicando as operações de projeção (π) apropriadas a FUNC_PROJ.

Suponha que usamos FUNC_PROJ1 e FUNC_LOCAL como relações da base em vez de FUNC_PROJ. Isso produz um projeto de esquema particularmente ruim, pois não podemos recuperar a informação que havia originalmente em FUNC_PROJ de FUNC_PROJ1 e FUNC_LOCAL. Se tentarmos uma operação de junção natural sobre FUNC_PROJ1 e FUNC_LOCAL, o resultado produz muito mais tuplas do que o conjunto original de tuplas em FUNC_PROJ.

Diretriz 4: Projete esquemas de relação de modo que possam ser unidos com condições de igualdade sobre os atributos que são pares relacionados corretamente (chave primária, chave estrangeira) de um modo que garanta que nenhuma tupla falsa será gerada. Evite relações com atributos correspondentes que não sejam combinações (chave estrangeira, chave primária), pois a junção sobre tais atributos pode produzir tuplas falsas.

The diagram illustrates three tables related to the FUNC_PROJ relation from Figure 8.3:

- (a) FUNC_LOCAL**: A table with attributes Fnome and Projlocal. A dependency arrow labeled "Cpf" points from Projlocal to Fnome.
- (b) FUNC_PROJ**: A table with attributes Cpf, Projnumero, Horas, Projnome, and Projlocal. A dependency arrow labeled "Cpf" points from Projlocal to Cpf.
- (c) FUNC_LOCAL**: A detailed view of the FUNC_LOCAL table showing data for employees and their project locations.

FUNC_LOCAL	
Fnome	Projlocal
Silva, Jolio B.	Santo André
Silva, Jolio B.	Itu
Lima, Ronald K.	São Paulo
Leite, Joice A.	Santo André
Leite, Joice A.	Itu
Wong, Fernando T.	Itu
Wong, Fernando T.	São Paulo
Wong, Fernando T.	Mauá
Zelaysa, Alice J.	Mauá
Pereira, André V.	Mauá
Souza, Jennifer S.	Mauá
Souza, Jennifer S.	São Paulo
Bravo, Jorge E.	São Paulo

FUNC_PROJ				
Cpf	Projnumero	Horas	Projnome	Projlocalizacao
12345678906	1	32,5	ProdutoX	Santo André
12345678906	2	7,5	ProdutoY	Itu
66688444476	3	40,0	ProdutoZ	São Paulo
45345345376	1	20,0	ProdutoU	Santo André
45345345376	2	20,0	ProdutoV	Itu
33344555587	2	10,0	ProdutoZ	Itu
33344555587	3	10,0	ProdutoZ	São Paulo
33344555587	10	10,0	Computadorização	Mauá
33344555587	20	10,0	Reorganização	São Paulo
99988777767	30	30,0	Novosbenefícios	Mauá
99988777767	10	10,0	Computadorização	Mauá
98765432168	10	35,0	Computadorização	Mauá
98765432168	30	5,0	Novosbenefícios	Mauá
98765432168	30	20,0	Novosbenefícios	Mauá
98798798733	20	15,0	Reorganização	São Paulo
88866555576	20	NULL	Reorganização	São Paulo

Figura 8.5: Projeto particularmente fraco para a relação FUNC_PROJ da Figura 8.3.

8.2 Dependência Funcional

Vimos as medidas informais do projeto de banco de dados. Agora, vamos falar de uma ferramenta formal para a análise de esquemas relacionais, que nos permite detectar e descrever alguns dos problemas mencionados em termos precisos. O conceito mais importante na teoria de projeto de esquema relacional é o de uma dependência funcional. Uma **dependência funcional** é uma restrição entre dois conjuntos de atributos do banco de dados.

Suponha que o esquema de relação R possua n atributos A_1, \dots, A_n . Uma dependência funcional (DF), representada por $X \rightarrow Y$ entre dois conjuntos de atributos x e y que são subconjuntos de R, especifica uma restrição nas tuplas de R.

Se $t_1[X] = t_2[X]$ então $t_1[Y] = t_2[Y]$. Isto significa que os valores dos atributos de Y dependem dos valores dos atributos de X.

$X \rightarrow Y$: lê-se y é funcionalmente dependente de x.

Exemplo: $NSS \rightarrow \{Nome, Nasc, End\}$

$$t_1[NSS] = t_2[NSS] \Rightarrow t_1[Nome, Nasc, End] = t_2[Nome, Nasc, End]$$

A DF deve ser explicitamente definida por alguém que conheça a semântica dos atributos.

Uma dependência funcional é uma propriedade da **semântica ou significado dos atributos**.

Considere o esquema de relação FUNC_PROJ da Figura 8.3b. Pela semântica dos atributos e da relação, sabemos que as seguintes dependências funcionais devem ser mantidas:

- a $Cpf \rightarrow Fnome$
- b $Projnumero \rightarrow \{Projnome, Projlocal\}$
- c $\{Cpf, Projnumero\} \rightarrow Horas$

Essas dependências funcionais especificam que (a) o valor do número do Cadastro de Pessoa Física (Cpf) de um funcionário determina exclusivamente o nome do funcionário (Fnome), (b) o valor do número de um projeto (Projnumero) determina exclusivamente o nome do projeto (Projnome) e seu local (Projlocal) e (c) uma combinação de valores de Cpf e Projnumero determina exclusivamente o número de horas que o funcionário costuma trabalhar no projeto por semana (Horas). De outra

maneira, podemos dizer que Fnome depende funcionalmente de Cpf, ou, dado um valor de Cpf, sabemos o valor de Fnome, e assim por diante.

Uma dependência funcional é uma propriedade da tabela R, e não um estado válido e específico de R. Por tanto, uma DF não pode ser deduzida automaticamente por determinada extensão de relação r, mas deve ser definida de maneira explícita por alguém que conhece a semântica (significado) dos atributos de R. Por exemplo, a Figura 8.6 mostra um estado da tabela ENSINA. Num primeiro momento podemos pensar que Texto → Disciplina, não podemos confirmar isso a menos que saibamos que é verdadeiro para *todos os estados legais possíveis* de ENSINA. É suficiente demonstrar um único contraexemplo para refutar uma dependência funcional. Por exemplo, como 'Silva' leciona tanto 'Estruturas de Dados' e 'Gerenciamento de Dados', podemos concluir que o Professor *não* determina funcionalmente a Disciplina.

ENSINA		
Professor	Disciplina	Texto
Silva	Estruturas de Dados	Bartram
Silva	Gerenciamento de Dados	Martin
Neto	Compiladores	Hoffman
Braga	Estruturas de Dados	Horowitz

Figura 8.6: Um estado da tabela ENSINA com uma possível dependência funcional TEXTO → DISCIPLINA.

8.3 Normalização

Agora que temos o conceito de dependência funcional, podemos usá-lo para especificar alguns aspectos da semântica dos esquemas de relação. Consideramos que um conjunto de dependências funcionais é dado para cada relação, e que cada relação tem uma chave primária designada. Essa informação combinada com os testes (condições) para formas normais controla o processo de normalização para o projeto do esquema relacional.

O processo de normalização pode ser visto como sendo a eliminação de relações não satisfatórias, decompondo-as através da separação de seus atributos em relações menos complexas, mas que atendem a propriedades desejadas.

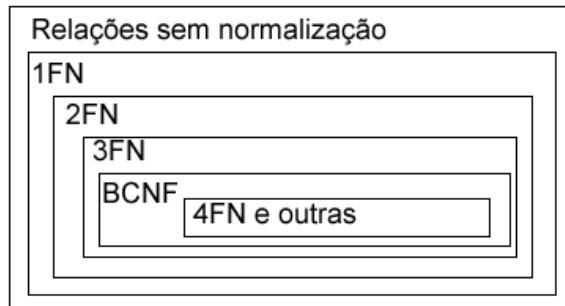


Figura 8.7: Esquema de relação entre as Formas Normais.

O processo de normalização consiste em realizar uma bateria de testes para certificar em que forma normal a relação se encontra. Estas formas normais são baseadas em DF dos atributos da

relação.

Inicialmente, Codd propôs três formas normais, que ele chamou de primeira, segunda e terceira forma normal. Uma definição mais forte da 3FN - chamada Forma Normal de Boyce-Codd (FNBC) - foi proposta posteriormente por Boyce e Codd. Todas essas formas estão baseadas em uma única ferramenta analítica: as dependências funcionais entre os atributos de uma relação. Depois, uma quarta (4FN) e uma quinta (5FN) formas normais foram propostas com base nos conceitos de dependências multivaloradas e dependências de junção.

8.3.1 Primeira Forma Normal (1FN)

Definição: Uma relação está em 1FN se todos os atributos forem atômicos e monovalorados, ou seja, não possuem valores que formam atributos compostos.



Figura 8.8: Exemplo de esquema de relação na 1FN.

Historicamente, a primeira forma normal foi definida para reprovar atributos multivalorados, atributos compostos e suas combinações. Ela afirma que o domínio de um atributo deve incluir apenas valores atômicos (simples, indivisíveis) e que o valor de qualquer atributo em uma tupla deve ser único valor do domínio dessa atributo. Logo, 1FN reprova ter um conjunto de valores, uma tupla de valores ou uma combinação de ambos como um valor de atributo para uma única tupla. Em outras palavras, 1FN reprova *relações dentro de relações ou relações como valores de atributos dentro de tuplas*. Os únicos valores permitidos pela 1FN são os valores **atômicos** (ou **indivisíveis**).

(a)

DEPARTAMENTO

Dnome	Dnumero	Cpf_gerente	Dlocal
Pesquisa	5	33344555587	Santo André, Itu, São Paulo
Administração	4	98765432168	Mauá
Matriz	1	88866555576	São Paulo

(b)

DEPARTAMENTO

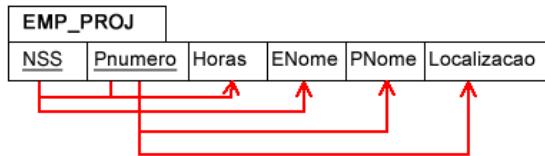
Dnome	Dnumero	Cpf_gerente	Dlocal
Pesquisa	5	33344555587	Santo André, Itu, São Paulo
Administração	4	98765432168	Mauá
Matriz	1	88866555576	São Paulo

Figura 8.9: Normalização na 1FN. (a) Exemplo de relação que não está na 1FN. (b) Exemplo de estado da tabela DEPARTAMENTO

8.3.2 Segunda Forma Normal (2FN)

Usa o conceito de *Dependência Funcional Total*. Uma DF $x \rightarrow y$ é total, se remover um atributo de x e a DF deixa de existir; uma DF é parcial se remover um atributo A de x e a DF continua a existir.

Definição: Uma relação R está na 2FN se estiver na 1FN e todos atributo que não pertença a alguma de suas chaves for totalmente dependente da sua chave primária, ou seja para que uma relação esteja na 2FN é preciso que esteja em 1FN e todos os atributos que não são chaves dependam da chave primária.



$$\{NSS, Pnumero\} \rightarrow \{Horas\}$$

$$\{NSS\} = \{Enome\}$$

$$\{Pnumero\} = \{Pnome, Localizacão\}$$

No esquema acima, $\{NSS, Pnumero\} \rightarrow Horas$ é uma dependência total (nem $NSS \rightarrow Horas$ nem $Pnumero \rightarrow Horas$ se mantêm). Contudo, a dependência $\{NSS, Pnumero\} \rightarrow Enome$ é parcial porque $NSS \rightarrow Enome$ se mantém.

O teste para 2FN envolve verificar se os atributos do lado esquerdo das dependências funcionais fazem parte da chave primária. Se a chave primária contiver um único atributo, a necessidade do teste não se aplica. A relação EMP_PROJ está na 1FN, mas não na 2FN. O atributo não primário Enome viola a 2FN em razão da DF.

Para passar para a 2FN, temos:

A relação original deixa de existir, ficando no lugar as três outras relações que satisfazem as condições para 2FN

$$\left\{ \begin{array}{l} R_1(NSS, Pnumero, Horas) \\ R_2(NSS, Enome) \\ R_3(Pnumero, Pnome, Localizacão) \end{array} \right.$$

8.3.3 Terceira Forma Normal (3FN)

Usa o conceito de *Dependência Funcional Transitiva*. Uma DF $x \rightarrow y$ é transitiva em um esquema R se existir um conjunto de atributos Z que não é um subconjunto de chaves de R e as $DF_s x \rightarrow z, z \rightarrow y$ são válidas.

Definição: Uma relação está na 3FN se estiver na 2FN e nenhum atributo que não pertença a alguma de suas chaves for transitivamente dependente da sua chave primária, ou seja, para que esteja na 3FN é preciso estar na 2FN e se todo atributo, que não pertença a alguma chave, for não dependente de algum atributo, que também não pertença a alguma chave.

Exemplo:

An relação EMP_DEPT, abaixo, a dependência $NSS \rightarrow NSSGER$ é transitiva por meio de Dnumero, pois ambas as dependências $NSS \rightarrow Dnumero$ e $Dnumero \rightarrow NSSGER$ se mantêm e Dnumero não é nem uma chave por si só nem um subconjunto da chave de EMP_DEPT.

EMP_DEPT					
NSS	Enome	Nasc	Dnumero	DNome	NSSGER

The diagram shows a horizontal red bracket under the first three columns (NSS, Enome, Nasc) with four red arrows pointing upwards to the corresponding columns in the second row. Another horizontal red bracket under the fourth column (Dnumero) has two red arrows pointing upwards to the fifth and sixth columns (DNome and NSSGER).

$$\{ \text{NSS} \} \rightarrow \{ \text{Enome}, \text{Nasc}, \text{Dnumero} \}$$

$$\{ \text{P} \text{Dnumero} \} = \{ \text{Dnome}, \text{NSSGER} \}$$

A dependência funcional (DF) $\{ \text{NSS} \} \rightarrow \{ \text{NSSGER} \}$ é uma DF transitiva via Dnumero. Verifica-se que a DF $\{ \text{Dnumero} \} \rightarrow \{ \text{NSSGER}, \text{Dnome} \}$ é indesejável uma vez que Dnumero não é chave primária da relação.

Para passar para 3FN:

$$\left\{ \begin{array}{l} R_1(\underline{\text{NSS}}, \text{Pnumero}, \text{Nasc}, \text{Dnumero}) \\ R_2(\underline{\text{Dnumero}}, \text{Dnome}, \text{NSSGER}) \end{array} \right.$$

8.3.4 Forma Normal Boyce-Codd

Verificou-se alguns inconvenientes na 3FN, tais como: não tratava adequadamente o caso de uma relação que tem duas ou mais chaves candidatas compostas e com superposição (com um atributo comum).

Para analisar a FN de Boyce-Codd é preciso considerar as situações onde existem duas chaves candidatas possíveis que se superpõe e se compõe para uma determinada relação. Para relações que tenham apenas uma chave candidata a FNBC reduz-se a 3FN.

No processo de normalização, essa forma normal deve ser aplicada às tabelas em 3FN que possuam mais de uma chave candidata (lembre-se de que a chave primária é também uma chave candidata), onde pelo menos uma delas seja composta e onde haja superposição entre elas.

Para simplificar, definimos que uma tabela está em FNBC se e somente se todos os determinantes são chaves candidatas. Ou seja, se houver algum atributo que seja determinado por outro(s) atributo(s) que não é (sejam) uma chave candidata, não estamos na FNBC. A solução é levar esses atributos para outra tabela, utilizando o conceito de decomposição sem perdas.

Vejamos um exemplo. Seja a relação Filmes1 com as suas dependências funcionais.

Filmes1(nome, ano, duração, nomeEstudio, localEstudio)

$\{ \text{nome}, \text{ano} \} \rightarrow \text{duração}$

$\{ \text{nome}, \text{ano} \} \rightarrow \text{nomeEstudio}$

$\{ \text{nome}, \text{ano} \} \rightarrow \text{localEstudio}$

$\{ \text{nome} | \text{Estudio} \} \rightarrow \text{localEstudio}$

A Figura 8.10 mostra uma possível instância para a tabela Filmes1. Podemos observar a presença de redundância, pois o local de um estúdio aparece repetido várias vezes. Pode dar origem a anomalias de "updates" e de "deletes".

A chave é $\{ \text{nome}, \text{ano} \}$ porque $\rightarrow \{ \text{duração}, \text{nomeEstudio}, \text{localEstudio} \}$

Filmes1 não está na FNBC porque $\{ \text{nomeEstudio} \}$ não é superchave de Filmes. $\{ \text{nomeEstudio} \} \rightarrow \{ \text{localEstudio} \}$ viola a condição FNBC.

Podemos decompor Filmes1 em:

- Filmes2(nome, ano, duração, nomeEstudio)
- Filmes3 (nomeEstudio, localEstudio)

nome	ano	duração	nomeEstúdio	moradaEstúdio
Star Wars	1977	124	Fox	10 Elm St., Los Angeles
Empire Strikes Back	1980	143	Fox	10 Elm St., Los Angeles
Gone With the Wind	1939	181	Paramount	44 Pine St., Los Angeles
Lion King	1994	124	Disney	56 Oak St., Los Angeles
Return of the Jedi	1983	165	Fox	10 Elm St., Los Angeles
Pocahontas	1995	115	Disney	56 Oak St., Los Angeles

Figura 8.10: Possível estado da tabela Filmes1

Para simplificar, definimos que uma tabela está em FNBC se e somente se todos os determinantes são chaves candidatas. Ou seja, se houver algum atributo que seja determinado por outro(s) atributo(s) que não é (sejam) uma chave candidata, não estamos na FNBC. A solução é levar esses atributos para outra tabela, utilizando o conceito de decomposição sem perdas.

8.4 Exercícios

1. Considere as seguintes relações e suas dependências funcionais.

- $\text{Peça} = \{\text{codigo, fornecedor, endereçoFornecedor, qtdEstoque, peso, valor}\}$
 $\text{codigo, fornecedor} \rightarrow \text{valor};$
 $\text{fornecedor} \rightarrow \text{endereçoFornecedor};$
 $\text{codigo} \rightarrow \text{peso, qtdEstoque};$
- $\text{Livro} = \{\text{ISBN, autor, instituicaoAutor, titulo, editora, tipo, preço}\}$
 $\text{ISBN} \rightarrow \text{autor, titulo, editora, tipo}$
 $\text{autor} \rightarrow \text{instituiçãoAutor}$
 $\text{tipo} \rightarrow \text{preço}$
 $\text{autor, titulo} \rightarrow \text{ISBN, editora}$

Para cada relação indique se as formas normais 1NF, 2NF e 3NF são atendidas. Normalize as relações para atender a cada uma destas formas.

2. Dados os formulários da Figura 8.11:

The image shows two rectangular forms side-by-side. The left form is titled 'Livros & Cia.' and 'Ped. Num: ____'. It has fields for 'Nome cliente:', 'End.', and 'End. Entrega:'. Below these is a table titled 'Livros solicitados' with columns 'Cód.', 'Quant.', 'Preço unit.', and 'Total'. At the bottom is a field 'Vendedor: ____'. The right form is also titled 'Livros & Cia.' and 'Ficha de livro'. It has fields for 'Código:', 'Título:', 'Autores:', and 'Preço R\$: ____'.

Figura 8.11: Formulários de solicitação e ficha de livros

- A partir dos formulários defina uma relação denominada “Pedido” e outra denominada “Livro”.
- Defina as dependências funcionais para os dados das relações criadas.
- A partir das DFs definidas no item b, normalize os esquemas até a 3FN, apresentando todo o desenvolvimento

3. Considere a relação

`emp_proj (nro_emp, nome_emp, {projeto (nro_proj, nome_proj) })`

{ } indica que o atributo projeto é multivalorado;

{projeto () } indica os atributos componentes do atributo multivalorado projeto.

Como normalizá-la para a 1FN?

4. Seja o esquema de relação cliente, Figura 8.12, e um estado atual de dados.

- cliente (nro_cli, nome, {end_entrega})

nro_cli	nome	end_entrega
124	João dos Santos	Rua 10, 1024 Rua 24, 1356
311	José Ferreira Neves	Rua 46, 1344 Rua 98, 4456

Figura 8.12: Formulários de solicitação e ficha de livros

- Esta tabela não está na 1FN. Por que?
- Normalize a tabela cliente até a 1FN, explicando todo o desenvolvimento.

5. O esquema de relação Pedido, abaixo, não está na 2FN. Porque?

`Pedido (nro-pedido, data, nro-peça, descrição, qtdade_comprada, preço_cotado)`

`nro-pedido → data`

`nro-peça → descrição`

$\{ \text{nro-pedido}, \text{nro-peça} \} \rightarrow \{ \text{qtdade_comprada}, \text{preço_cotado} \}$

Normalize-o, indicando todo o desenvolvimento.

6. Sejam as relações abaixo e as DFs identificadas pelo desenvolvedor.

`Ministra={Professor, Sigla, LivroTexto, LivroExerc}`

`Turma={NúmeroT, Sigla, Sala, No.Horas}`

DFs identificadas pelo desenvolvedor:

$\{ \text{Professor}, \text{Sigla} \} \rightarrow \{ \text{LivroTexto} \};$

$\{ \text{NúmeroT}, \text{Sigla} \} \rightarrow \{ \text{Sala} \};$

`Sigla → No.Horas;`

`LivroTexto → → LivroExerc.`

Indique em que forma normal cada tabela se encontra e normalize-a, quando necessário, indicando todo o desenvolvimento.

7. Nos exercícios seguintes, normalize as relações de forma que todas as relações resultantes estejam na forma normal mais restrita. Considere a 1FN, a 2FN e a 3FN. Para cada FN:

- Se necessário, identifique quais as dependências funcionais que se aplicam sobre R;
- Identifique e justifique se R encontra-se ou não na forma normal em questão; e
- Caso R sendo analisada não se encontre na forma normal em questão, normalize-a, especificando as relações originadas.

(a) `vendedor (nro_vend, nome_vend, {cliente (nro_cli, nome_cli)})`

- As seguintes dependências funcionais devem ser garantidas na normalização:
 - $nro_vend \rightarrow nome_vend$;
 - $nro_cli \rightarrow nome_cli$
- Observação: considere que um vendedor pode atender diversos clientes, e um cliente pode ser atendido por diversos vendedores.

(b) `aluno (nro_aluno, cod_dept, nome_dept, sigla_dept, cod_orient, nome_orient, fone_orient, cod_curso)`

- As seguintes dependências funcionais devem ser garantidas na normalização:
 - $cod_dept \rightarrow \{ nome_dept, sigla_dept \}$;
 - $cod_orient \rightarrow \{ nome_orient, fone_orient \}$;
 - $nro_aluno \rightarrow \{ cod_dept, cod_orient, cod_curso \}$;
- Observações adicionais:
 - um aluno somente pode estar associado a um departamento;
 - um aluno cursa apenas um único curso;
 - um aluno somente pode ser orientado por um único orientador.

8. Considere os seguintes requisitos para um banco de dados de universidade, usado para registrar os históricos dos alunos:

- (a) A universidade registra o nome de cada aluno (Anome), o número do aluno (Anum), o número do Cadastro de Pessoa Física (CPF), o endereço moradia (Aendereco_mora) e o número do telefone (Atelefone_mora), endereço permanente (Aendereco_fixo) e telefone (Atelefone_fixo), data de nascimento (Datanasc), sexo (Sexo), tipo_aluno ('calouro', 'veterano', 'graduado'), departamento principal (Dep_princ), departamento secundário (Dep_sec) (se houver) e programa de título (Titulacao) ('bacharel', 'mestrado', ..., 'doutorado'). Tanto CPF quanto o número do aluno possuem valores únicos para cada um.
- (b) Cada departamento é descrito por um nome (Dnome), código de departamento (Dcodigo), número de escritório (Descriptorio), telefone de escritório (Dtelefone) e faculdade (Dfaculdade). Tanto o nome quanto o código possuem valores únicos para cada departamento.
- (c) Cada disciplina tem um nome (Dnome), descrição (Ddesc), número (Dnum), número de horas semestrais (Credito), nível (Nivel) e departamento de oferta (Num_dep). O número da disciplina é único para cada curso.
- (d) Cada turma tem um professor (Unome), semestre (Semestre), ano (Ano), disciplina (Disciplina_turma) e número de turma (Num_turma). O número de turma distingue diferentes turmas da mesma disciplina que não são lecionadas durante o mesmo semestre/ano; seus valores são 1, 2, 3, ..., até o número total de turmas lecionadas durante cada semestre.

- (e) Um registro de nota refere-se a um aluno (Anum), uma turma em particular, e a uma nota (Nota). Crie um esquema de banco de dados relacional para essa aplicação de banco de dados. Primeiro, identifique todas as dependências funcionais que devem ser mantidas entre os atributos. Depois, projete no MySQL Workbench esquemas de relação para o banco de dados que estejam, cada uma, na 3FN.

9 Referências Bibliográficas

Elmasri, Ramez; Navathe, Shamkant B. **Fundamentals of database systems**. 6th ed. Pearson. 1172p. 2011.

MySQL. **MySQL 5.7 Reference Manual**. Disponível em: <<https://dev.mysql.com/doc/refman/5.7/en/>> Acesso em: Dez/2022.

Takahashi, Mana; Azuma, Shoko; Trend-pro Co. **Guia Mangá de banco de dados**. Tradução Thaís Cristina Casson. São Paulo: Novatec Editora, 2009.

W3Schools. **SQL Tutorial**. Disponível em: <<https://www.w3schools.com/sql/>> Acesso em: Dez/2022.

Índice Remissivo

- abstração, 11, 15
 - de dados, 11, 15
- analista, 12
 - de dados, 12
 - de sistemas, 12
- aplicativos, 20
- aplicação, 11
- armazenamento, 16, 17
 - arquivos, 16
 - físico, 17
- arquitetura, 16, 20
 - centralizada, 20
 - cliente servidor, 20
- arquitetura de três esquemas, 16, 17
- atomicidade, 11
- atributo, 15
- autodescrição, 11, 16
- backup, 13
- banco, 9
- banco de dados, 11–16, 20
 - aplicações de, 14
- Big Data, 13
- commit, 20
- compartilhamento, 11, 13
 - de dados, 11, 13
- concorrência, 13
 - controle, 13
 - controle de, 11
- consistência, 15
- consulta, 11, 19, 20
 - linguagem de, 20
- dados, 9, 12–14, 16
 - relacionados, 9
- DBA, 11, 12, 18
- DDL, 18
- descrição
 - de dados, 15
 - do banco de dados, 16
- desenvolvedor, 12
- diagrama, 16, 17, 19
 - de esquema, 16
- disponibilidade, 11
- DML, 19
- entidade, 15, 17
- esquema, 15, 16, 18
 - alteração, 20
 - alteração de, 17
 - conceitual, 17, 18
 - de banco de dados, 15, 16
 - evolução do, 16
 - externo, 16, 17
 - interno, 17, 18
 - níveis de, 16, 17
- estado, 16
 - atual, 16
 - de banco de dados, 16
 - inicial, 16
 - válido, 16
- estrutura, 15
 - de registro, 16
 - lógica, 18
- independência, 11, 16
 - de dados, 17–19
 - física, 18
 - lógica, 17, 18
- instância, 16
- integridade, 13
- interface, 19
 - baseada em formulários, 19
 - baseada em menus, 19
 - de linguagem natural, 20
 - do SGBD, 19
 - gráfica, 19
 - para administrador de banco de dados, 20
 - parametrizável, 20
- isolamento, 11, 16
- mapeamento, 18
- metadados, 11, 16
- modelo, 11, 12, 15
 - conceitual, 11
- modelo de dados, 15, 16
 - baseados em registro, 16
 - conceitual, 15, 16
 - de implementação, 15–17
 - físico, 15–17
 - relacional, 16

modelo entidade relacionamento, 15
mundo real, 9, 15

nível conceitual, 17
nível externo, 16
nível interno, 17

objeto, 15
OLTP, 11
operações, 19
 alteração, 19
 inserção, 19
 recuperação, 19
 remoção, 19

processamento, 11, 20
 de transação, 11

projetista, 12

projeto de banco de dados, 16

propriedades
 da transação, 11

relacionamento, 15, 17

requisito, 12, 14

restore, 13

restrições, 13, 17
 de esquema, 16

rollback, 20

SBD, 10, 11

SDL, 18

segurança, 12

semântica, 15

SGBD, 9–14, 16, 17, 19–21
 catálogo, 16

sistema de banco de dados, 10, 11, 17

SQL, 12

stored procedure, 20

tabela, 18

transação, 11, 20

trigger, 20

usuário, 9, 10, 13–16, 19
 final, 19
 multusuário, 11
 múltiplos, 14

VDL, 18

visões, 11, 16

 de dados, 11
 de usuário, 18

