



BANCO DE DADOS

Desenvolvimento
de aplicações

2ª edição

Denise Bandeira

Escola
Politécnica

COLEÇÃO
EAD
EDITORA UNISINOS



BANCO DE DADOS

Desenvolvimento
de aplicações

2ª edição

Denise Bandeira

Escola
Politécnica

COLEÇÃO

EAD

EDITORA UNISINOS

BANCO DE DADOS: DESENVOLVIMENTO DE APLICAÇÕES

DENISE BANDEIRA

2ª edição

Editora Unisinos, 2015

SUMÁRIO

Apresentação

Capítulo 1 – Introdução

Capítulo 2 – Modelo relacional

Capítulo 3 – Modelagem entidade-relacionamento

Capítulo 4 – Mapeamento de modelo entidade-relacionamento para modelo relacional

Capítulo 5 – Normalização

Capítulo 6 – Álgebra relacional

Capítulo 7 – SQL

Referências

Sobre a autora

Informações técnicas

APRESENTAÇÃO

Este livro apresenta os principais conceitos relacionados ao desenvolvimento do projeto de Bancos de Dados. No Capítulo 1 são apresentados os conceitos introdutórios que descrevem a área, passando pelos conceitos de banco de dados, sistema gerenciador de banco de dados, modelo de dados, modelagem de dados e projeto de banco de dados. O Capítulo 2 apresenta o Modelo Relacional, suas características estruturais e aspectos de integridade. Os aspectos relacionados à metodologia de Modelagem Entidade-Relacionamento são apresentados no Capítulo 3, sendo que as regras de mapeamento do modelo conceitual para o modelo relacional são apresentadas no Capítulo 4. O Capítulo 5 aborda outra técnica para elaboração e validação do projeto de lógico de bancos de dados, que é a Normalização. E, por fim, os Capítulos 6 e 7 são dedicados à linguagem para criação e manipulação de Bancos de Dados, a Álgebra Relacional e SQL.

Espera-se que este livro sirva como base para os estudos relacionados à área de Banco de Dados. Entretanto, é indispensável que os materiais indicados para a leitura complementar, as referências bibliográficas indicadas em cada capítulo, bem como a lista de referências bibliográficas final para serem consultadas.

Aautora.

CAPÍTULO 1

INTRODUÇÃO

Este capítulo apresenta os conceitos introdutórios da área de banco de dados, assim como uma contextualização em relação de que maneira os sistemas tratavam do armazenamento e gerenciamento dos dados antes e após o surgimento da tecnologia de banco de dados.

Durante as últimas décadas cresceu o emprego de bancos de dados como mecanismo para gerenciar as informações de qualquer tipo de organização, sejam escolas, fábricas, lojas de varejo etc. A quantidade de informações com que tais organizações lidam nos dias atuais gera uma situação em que a administração dessas organizações só é possível por meio da informatização. O emprego da **tecnologia de banco de dados** é o elemento da informática que permite o gerenciamento desses grandes volumes de dados. Fala-se em **tecnologia de banco de dados** porque se refere a um conjunto de recursos, contemplando desde aqueles que permitem o armazenamento dos dados propriamente ditos até o seu gerenciamento.

A Figura 1 mostra a representação dos dados em sistemas computacionais,¹ antes do conceito de banco de dados. No exemplo, são apresentados o caso de uma Universidade e os sistemas que realizam o gerenciamento de suas informações. Cada aplicação mantém seus dados de forma isolada, mesmo que as informações manipuladas possam ser compartilhadas entre todas elas.

Na situação apresentada no contexto da Figura 1, cada aplicação² da organização mantém o seu conjunto de dados. Nesse caso, a descrição dos dados fica dentro das aplicações e não existe compartilhamento deles entre as aplicações.

Os problemas advindos dessa situação são:

- Replicação do conjunto de dados armazenados no sistema: cada aplicação mantém os dados de que necessita, mesmo que sejam os mesmos dados que outra aplicação também precisa manipular. Com isso, os dados estão replicados, considerando o sistema como um todo;
- Dificil manutenção dos dados: o fato de os dados estarem replicados em locais distintos, sendo mantidos por diferentes aplicações, torna a tarefa de manutenção mais difícil, pois cada

alteração feita em uma das réplicas deveria ser feita também nas demais;

Figura 1 – Sistemas de uma Universidade sem uso de BD.
Fonte: elaborada pelo professor Ronaldo dos Santos Mello (UFSC).

Falta de padronização na definição dos dados: como os dados são definidos dentro de cada aplicação, pode ser que cada uma delas os defina de uma forma diferente. Por exemplo, uma determinada aplicação pode armazenar o endereço de uma pessoa como um conjunto de caracteres simplesmente e outra aplicação como um conjunto de elementos individuais, como rua, número, complemento e CEP separadamente.

Surge, então, a necessidade de manter-se uma melhor organização e gerência sobre os dados.

1.1 Banco de Dados

A definição de Banco de Dados (BD) é que atende a essa necessidade é: “Uma coleção de dados inter-relacionados logicamente, e suas descrições, projetados para atender às necessidades de informação das organizações” (CONNOLLY, 2010).

Com a aplicação desse conceito, é possível manter os dados armazenados de forma controlada e compartilhada entre as aplicações. Os dados são armazenados de forma independente das aplicações, ou seja, há uma padronização e independência na estruturação dos dados, ao invés de cada aplicação definir internamente como eles serão armazenados. E, por fim, as aplicações não se preocupam com a gerência dos dados.

Na Figura 2, os mesmos dados da aplicação são apresentados como exemplo para o caso de uma Universidade, mas atendendo ao conceito de banco de dados. As caixas, no desenho, representam o compartilhamento dos dados entre as aplicações.

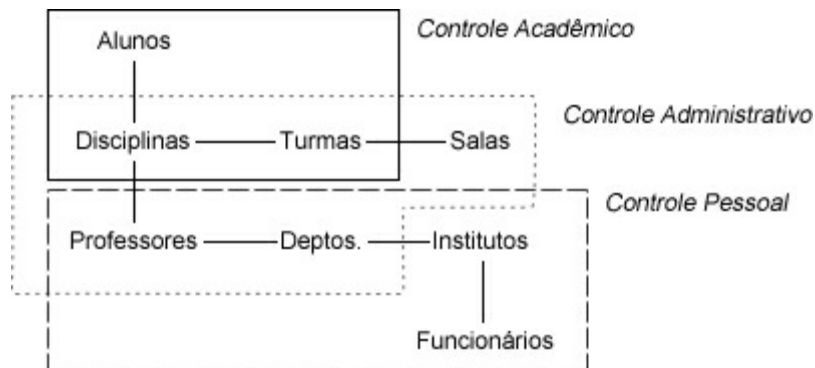


Figura 2 – Sistemas de uma Universidade com uso de BD.

Fonte: elaborada pelo professor Ronaldo dos Santos Mello (UFSC).

O gerenciamento do acesso e manutenção dos dados armazenados em um banco de dados é feito por um sistema chamado de Sistema Gerenciador de Banco de Dados (SGBD).

1.2 Sistema Gerenciador de Banco de Dados (SGBD)

Segundo (SILBERSCHATZ, 2006), um Sistema Gerenciador de Banco de Dados (SGBD) é um conjunto de programas que gerencia o acesso aos dados armazenados em Banco de Dados (BD).

Pode-se dizer que o SGBD possibilita diferentes visões de um banco de dados. O nível Físico é o nível mais baixo de abstração e define como os dados estão armazenados. O nível Conceitual (ou Lógico) define os dados que são armazenados no banco de dados e o relacionamento entre eles. E o nível de Visão (ou Externo) permite a visualização de apenas alguns detalhes dos dados e o usuário (aplicação) tem acesso de acordo com a necessidade. A Figura 3, abaixo, apresenta os elementos que compõem um sistema de banco de dados simplificado, considerando os três níveis explicitados anteriormente.

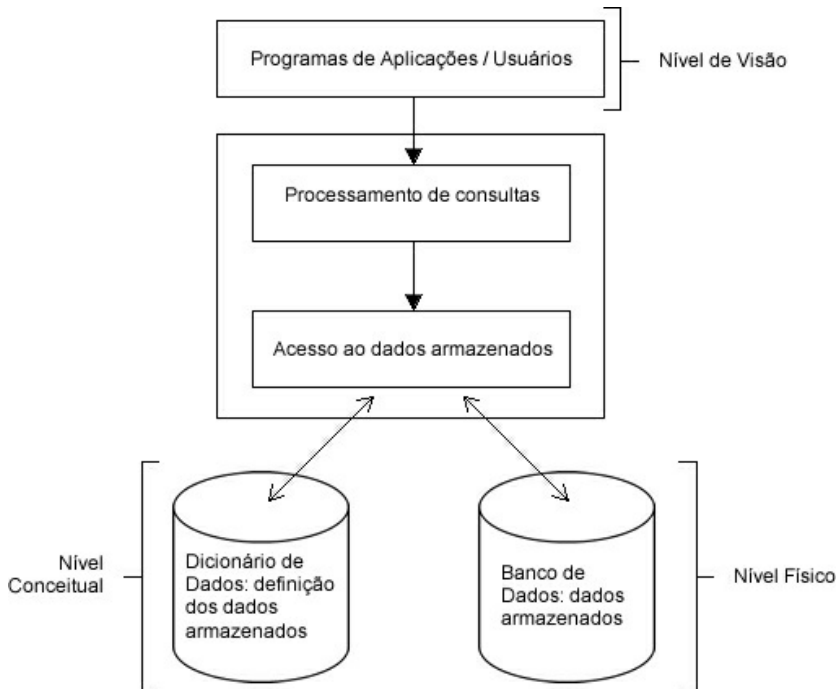


Figura 3 – Elementos de um sistema de banco de dados simplificado.
Fonte: ELMASRI, 2005.

1.3 Funcionalidades de um SGBD

Um SGBD, para permitir a interação entre os programas de aplicação/usuários e o banco de dados, deve possuir as seguintes funcionalidades básicas:

- Métodos que permitam a **definição dos dados e os métodos de acesso a eles**. Usualmente isso feito por meio de uma Linguagem de Definição de Dados (*DDL – Data Definition Language*). A linguagem padrão para interação com um banco de dados é SQL (*Structured Query Language*). DDL é a porção de SQL, ou seja, o conjunto de instruções de SQL que permite fazer a

definição dos dados que ficarão armazenados no banco de dados. A linguagem SQL será vista em detalhes no capítulo 6 deste livro. O resultado produzido pela definição é o que se chama de esquema do banco de dados e fica armazenado no Dicionário de Dados;

- Métodos que permitam a **manipulação dos dados armazenados** no banco de dados. Essa funcionalidade é realizada por meio de uma Linguagem de Manipulação de Dados (*DML - Data Manipulation Language*), que é a porção de SQL que permite inserções de novos dados no banco de dados, remoções e alterações;
- Mecanismos para realizar um **processamento eficaz de consultas**. O volume de dados em banco de dados é muito grande, por isso, faz-se necessário que o SGBD forneça mecanismos eficientes para processar e recuperar os dados;
- Mecanismos para controle e gerenciamento dos dados armazenados no banco de dados. Tais mecanismos podem contemplar: **restrições de integridade**, ou seja, regras que são definidas sobre os dados para regular as operações que podem ser realizadas sobre eles; **controle de acesso**, garantindo que somente usuários autorizados tenham acesso e direito à manipulação de porções específicas do banco de dados; **controle de concorrência**, permitindo que os dados sejam compartilhados entre as aplicações, mas garantindo que nenhum erro seja gerado decorrente desse compartilhamento; **sistemas de recuperação de falhas**, garantindo a recuperação do banco de dados após a ocorrência de qualquer falha que venha a interromper o funcionamento normal do sistema.

1.4 Usuários do SGBD

Um SGBD é um sistema bastante complexo e exige que uma série de tarefas específicas sejam realizadas para permitir tanto a configuração do próprio sistema, quanto a criação e manutenção dos bancos de dados por ele gerenciados. Para que as tarefas sejam realizadas de maneira satisfatória, os atores e papéis envolvidos são os seguintes:

- Administrador do Banco de Dados (*DBA – Database Administrator*): controle de diversas funcionalidades do SGBD

- definição e modificação do esquema do banco de dados;
 - definição e modificação da estrutura de armazenamento e métodos de acesso aos dados;
 - concessões de autorização de acesso;
 - especificação de restrições de integridade;
 - definição de estratégias de recuperação de falhas.
- Projetistas do Banco de Dados, responsáveis tanto pelo projeto conceitual/lógico³ quanto projeto físico⁴ do banco de dados.
 - Desenvolvedores das aplicações, responsáveis por desenvolver os programas que, por meio do SGBD, realizam o acesso e manipulação dos dados armazenados no banco de dados.
 - Usuários finais, que interagem direto com os dados do banco de dados, sejam por meio de interfaces de programas desenvolvidos especificamente para esse uso ou, no caso de usuários especializados, por meio da interface do próprio SGBD.

1.5 Principais fases do projeto de um banco de dados

O projeto de um banco de dados envolve diversas etapas, as quais estão descritas na Figura 4. O processo todo inicia pelo levantamento dos **requisitos de dados** daquilo que se deseja que conste no banco de dados. De posse desses requisitos, inicia-se a fase de construção do **projeto do banco de dados**. Entretanto, em paralelo ao levantamento dos requisitos de dados, é importante também realizar o levantamento dos **requisitos funcionais**, que darão origem ao **projeto dos programas de aplicação** que irão atuar sobre o banco de dados. Para compreender melhor todas as etapas apresentadas na Figura 4, são apresentados alguns conceitos a seguir.

os dados são organizados nas estruturas de armazenamento. Os modelos de dados mais conhecidos são: modelo hierárquico, modelo em redes, modelo relacional e modelos pós-relacionais (orientado a objetos, objeto/relacional e semiestruturados). A Figura 5 apresenta a evolução e principais características dos modelos de dados e SGBDs que os suportam.

Período	Modelo	Características
Início da década de 60	Sistemas de arquivos	Precusores dos SGBDs. Abordagem descentralizada: cada departamento armazenava e controlava seus próprios dados.
Metade da década de 60	Modelos de dados hierárquicos e em redes	Representam a primeira geração de SGBDs. O principal sistema que segue o modelo hierárquico é o IMS da IBM e o principal sistema que segue o modelo em redes é o IDMS/R da Computer Associates. Esses tipos de sistemas falhavam muito em relação à independência dos dados e, por isso, requeriam que programas complexos fossem desenvolvidos para processar os dados.
1970	Modelo Relacional	Publicação do trabalho original de E. F. Codd, intitulado “ <i>A relational model of data for large shared data Banks</i> ”, que declara as fraquezas dos SGBDs de primeira geração.
Década de 70	Protótipos de SGBDs Relacionais (SGBDRs)	Durante este período, dois principais protótipos surgiram: o projeto Ingres da Universidade da Califórnia, em Berkeley (iniciado em 1970) e o projeto System R do laboratório de pesquisa da IBM, em San José (iniciado em 1974), o qual desencadeou o desenvolvimento da linguagem SQL.

1976	Modelo ER foi proposto	Publicação do artigo de Peter Chen, intitulado “ <i>The Entity-Relationship model – Toward a unified view of data</i> ”. A modelagem ER torna-se um componente significativo na metodologia do projeto de banco de dados.
1979	SGBDs Comerciais aparecem	Sistemas como Oracle, Ingres e DB2 começam a aparecer. Eles representam a segunda geração dos SGBDs.
1987	Padrão SQL (ISO)	SQL é padronizada pela ISSO (<i>International Standards Organization</i>).
1990s	SGBDOO e SGBDOR aparecem	Começam a aparecer os Sistemas Gerenciados de Bancos de Dados Orientados a Objetos e Sistemas Gerenciados de Bancos de Dados Objeto Relacionais (Oracle 8 apresenta o conceito de objetos em 1997).

Figura 5 – Modelos e tipos de SGBDs e desenvolvimento histórico.

Fonte: CONNOLLY, 2010.

1.7 Modelagem de dados

É um método que permite modelar ou ilustrar o banco de dados. Normalmente é realizado por meio de uma forma gráfica, mas outras formas de comunicação são também desejáveis, pois os diagramas podem não ser totalmente compreensíveis. Diagramas ER (DER) é uma técnica gráfica que facilita o processo de modelagem. DER é considerado um modelo semântico para banco de dados, cujo objetivo é explicitar o significado dos dados. Esta técnica será apresentada mais adiante neste livro.

Quando se inicia a discussão a respeito do conteúdo de um banco de dados, a modelagem de dados ajuda na decisão de quais dados se relacionam com outros no nível conceitual. É preciso decidir qual o nível de abstração dos modelos que se está construindo, quanto mais longe se

fica dos detalhes concretos dos modelos lógicos de dados (hierárquico, relacional etc.) e dos detalhes físicos (tipos de dados etc.), mais fácil será para fazer mudanças no modelo e chegar a sua versão final.

1.8 Projeto do banco de dados

É o processo de criar um projeto para um banco de dados que irá dar suporte aos objetivos e operações de uma organização. Ele inclui o projeto conceitual, lógico e físico do banco de dados.

1.8.1 Projeto conceitual do banco de dados

É o processo de construção de um modelo com as informações usadas em uma organização, independentemente de todas as considerações físicas.

1.8.2 Projeto lógico do banco de dados

É o processo de construção de um modelo com as informações usadas em uma organização, baseado em um modelo de dados específico (relacional, por exemplo), mas independente de um SGBD específico e de outras considerações físicas.

1.8.3 Projeto físico do banco de dados

É o processo de produção de uma descrição da implementação do banco de dados em um meio de armazenamento; descreve as estruturas de armazenamento e os métodos de acesso usados para se obter acesso eficiente aos dados.

1.8.4 Mapeamento

É o processo de escolha de um modelo de dados e transformação do modelo conceitual em um modelo lógico/físico.

1.9 Recomendações para complementação de estudos referentes aos temas deste capítulo:

Codd, E. F. 1970. A relational model of data for large shared data banks. **Commun. ACM** 13, 6 (Jun. 1970), 377-387. Disponível em: <<http://doi.acm.org/10.1145/362384.362685>>. Acesso em: 23 jun. 2010.

Chen, P. P. 1976. The entity-relationship model—toward a unified view of data. **ACM Trans. Database Syst.** 1, 1 (Mar. 1976), 9-36. Disponível em: <<http://doi.acm.org/10.1145/320434.320440>>. Acesso em: 23 jun. 2010.

1.10 Resumo do capítulo

Neste capítulo foram apresentados os principais conceitos da área de banco de dados. É importante que os seguintes conceitos tenham sido assimilados, pois serão necessários para a compreensão dos conteúdos apresentados nos próximos capítulos:

- Banco de Dados (BD);
- Sistema Gerenciador de Banco de Dados (SGBD);
- Modelo de dados;
- Modelagem de dados;
- Projeto conceitual, lógico e físico de banco de dados.

¹ São sistemas que, por meio de *hardware* e *software* processam as informações de uma organização.

² Um problema que é tratado e resolvido por meio de um sistema computacional.

³ É o processo de construção de um modelo com as informações usadas em uma organização, independente do SGBD.

⁴ É o processo de produção de uma descrição da implementação do banco de dados

em um SGBD específico.

CAPÍTULO 2

MODELO RELACIONAL

Este capítulo apresenta as principais características do Modelo Relacional. Introduzido em 1970, como uma proposta simples e com uma base matemática muito bem definida, o Modelo Relacional tomou-se um dos modelos de dados com maior aceitação para criação de SGBDs comerciais, a partir da década de 80.

A proposta inicial do Modelo Relacional é de E. F. Codd, que trabalhava em um laboratório de pesquisa da IBM/San José, na Califórnia. O trabalho onde o Modelo Relacional foi primeiramente apresentado, intitulado “A Relational Model for Large Shared Data Banks”, é de 1970 (CODD, 1970).

O Modelo Relacional serviu de base para a criação da segunda geração de SGBDs porque envolvia conceitos simples e com uma base matemática muito sólida. Nesse modelo, os dados são estruturados logicamente em relações (tabelas), onde cada relação tem um nome, as relações são construídas por atributos (colunas) e cada tupla (linha) contém um valor por atributo. Veja um exemplo na Tabela 2.

Id_Funcionário	Nome_Funcionário	Departamento
100	João	Contábil
200	Pedro	Vendas
300	Maria	RH
400	Ana	Contábil
500	Luiz	Vendas

Figura 6 – Relação (tabela) Funcionário.

Fonte: elaborada pelo autor.

A Figura 6 apresenta a relação (tabela) Funcionário, que tem os atributos (colunas) Id_Funcionário, Nome_Funcionário e Departamento. Cada tupla (linha) da relação Funcionário tem valores específicos para cada um dos atributos.

2.1 Estrutura

O Modelo Relacional está baseado no conceito matemático de Relação. Uma relação $r(R)$ é uma relação matemática de grau n nos domínios D_1, D_2, \dots, D_n , que é um subconjunto do produto cartesiano dos domínios que definem R . O produto cartesiano contém todas as possíveis combinações de valores dos domínios D_1, D_2, \dots, D_n . A Figura 7 apresenta um exemplo, por meio de um Diagrama de Ven, de um produto cartesiano entre dois conjuntos, o conjunto **ID_Funcionário**, que tem como domínio os números de funcionários e o conjunto de **Departamento**, cujo domínio são os nomes de departamentos.

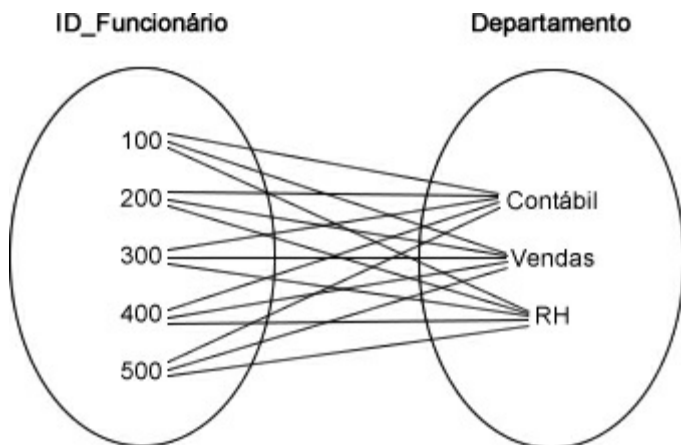


Figura 7 – Produto cartesiano entre os conjuntos de **Id_funcionário** e **departamento**.
Fonte: elaborado pelo autor.

Note que o diagrama da Figura 7 apresenta uma ligação de cada elemento do conjunto **Id_Funcionário** com todos os elementos de **Departamento**. Essa é uma forma diagramática de evidenciar um produto cartesiano.

A relação **Funcionário**, apresentada na forma de tabela na Figura 6, também pode ser descrita por meio de um Diagrama de Ven. Nesse caso, o diagrama irá apresentar somente as ligações que realmente acontecem entre **Id_Funcionário** e **Departamento**, conforme os dados já descritos na Figura 6. Veja:

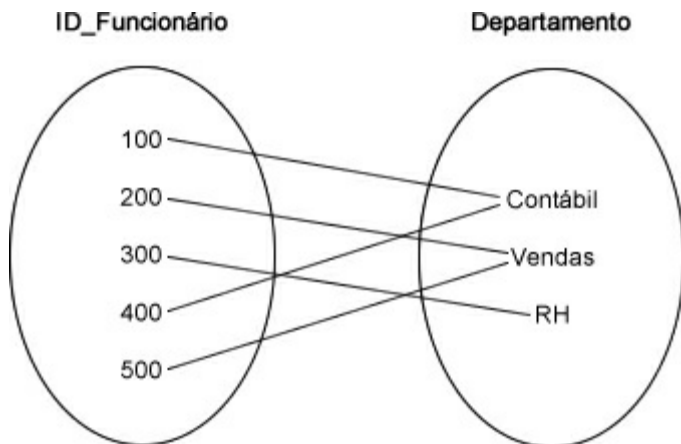


Figura 8 – Representação diagramática da relação Funcionário.

Fonte: elaborado pelo autor.

Note que o diagrama da Figura 8 apresenta apenas algumas ligações entre Id_Funcionário e Departamento, justamente para denotar (representar) um subconjunto das ligações do produto cartesiano mostrado na Figura 7. Observe também que foi omitido, nos diagramas das Figuras 7 e 8, o atributo Nome_Funcionário, apenas para facilitar a visualização.

Conforme exposto acima, os elementos estruturantes do Modelo Relacional são apresentados, resumidamente, na Figura 9.

Relação (tabela)	Conceito matemático que serve de base para o Modelo Relacional. Asua representação usual é uma tabela com duas dimensões.
Atributo (coluna)	Descreve características do elemento que está sendo representado por uma dada relação. Nome das colunas da tabela.
Domínio	Universo de valores permitido para um dado conjunto.
Tupla (linha)	Tem um componente para cada atributo da relação. Linhas da tabela.
Grau	Número de atributos (colunas) da relação (tabela).

Cardinalidade Número de tuplas (linhas) da relação (tabela).

Figura 9 – Elementos do modelo relacional.

Fonte: elaborada pelo autor.

2.2 Chaves

O conceito de chave é fundamental no Modelo Relacional e serve para identificar unicamente linhas de uma tabela (chave primária) e para estabelecer relações entre linhas de tabelas distintas do banco de dados (chave estrangeira). Uma chave pode ser constituída por um único atributo ou formada por um conjunto de atributos. Quando uma chave é formada por mais de um atributo ela é chamada de chave composta.

2.3 Chave primária

A chave primária é um atributo ou conjunto de atributos, cujos valores permitem identificar unicamente uma linha em uma tabela. Na tabela Funcionário, apresentada novamente na Figura 10, Id_Funcionário é o atributo que é a chave primária da tabela. Note que cada linha da tabela apresenta um valor diferente para o atributo Id_Funcionário.

Id_Funcionário	Nome_Funcionário	Departamento
100	João	Contábil
200	Pedro	Vendas
300	Maria	RH
400	Ana	Contábil
500	Luiz	Vendas

Figura 10 – Exemplo de tabela Funcionário e chave primária Id_Funcionário.

Fonte: elaborada pelo autor.

Quando a chave primária é uma chave composta por mais de um atributo da tabela, ela deve ser composta pelo conjunto mínimo de atributos que sejam suficientes para identificar unicamente as tuplas da tabela.

2.4 Chave estrangeira

A chave estrangeira é um atributo ou conjunto de atributos, que é a chave primária em outra tabela. A chave estrangeira é o mecanismo que permite estabelecer a relação entre diferentes tabelas do banco de dados. A Figura 11 apresenta um exemplo, onde, além da tabela Funcionário, está também a tabela Departamento, que é a tabela do Banco de Dados que descreve todos os Departamentos.

(a)			(b)	
Id_Funcionário	Nome_Funcionário	Departamento	Nome_Departamen	
100	João	Contábil	Contábil	
200	Pedro	Vendas	Vendas	
300	Maria	RH	RH	
400	Ana	Contábil		
500	Luiz	Vendas		

Figura 11 – (a) Tabela Funcionário e chave estrangeira Departamento, e (b) Tabela Departamento e chave primária Nome_Departamento.

Fonte: elaborada pelo autor.

Na tabela Funcionário, o atributo Departamento é chave estrangeira porque ele, justamente, é chave primária em outra tabela, na tabela Departamento, nesse caso. Note que o atributo tem nomes diferentes nas duas tabelas, isso não é um problema e nem é uma regra; eles podem tanto ter o mesmo nome como nomes diferentes.

2.5 O valor nulo

O valor nulo é atribuído a um atributo em uma tupla da tabela quando não existe um valor, no domínio daquele atributo, que possa ser atribuído em seu lugar. Por exemplo, se a tabela Funcionário possuísse o atributo CNH (Carteira Nacional de Habilitação), destinado a armazenar o número da carteira de motorista do funcionário, e um determinado funcionário não tivesse “tirado” ainda sua carteira de motorista, o valor **nulo** deveria ser armazenado nessa linha da tabela para denotar esse fato. Veja

como ficaria o conteúdo da tabela Figura 12.

Id_Funcionário	Nome_Funcionário	CNH	Departamento
100	João	10393857767	Contábil
200	Pedro	nulo	Vendas
300	Maria	94837277172	RH
400	Ana	84857575722	Contábil
500	Luiz	4757577860	Vendas

Figura 12 – Tabela Funcionário, com o atributo CNH com valor nulo para o funcionário “Pedro”.

Fonte: elaborada pelo autor.

Portanto, o valor nulo é empregado quando se quer denotar ausência de informação para um determinado atributo.

Sendo assim, os atributos que compõem a chave primária de uma tabela, como devem justamente permitir identificar as tuplas dessa tabela, então não podem conter o valor nulo. De outra forma, os atributos que compõem a chave estrangeira de uma tabela podem conter ou valores que estão presentes na tabela onde são a chave primária ou o valor nulo.

2.6 Integridade de entidade

Um dos aspectos de integridade do modelo relacional diz respeito à necessidade de que todos os elementos em um banco de dados possam ser identificados, pois dessa forma, poderiam ser acessados. Conforme visto anteriormente, o conceito que permite isso é o conceito de chave primária. Portanto, a integridade de entidade é uma restrição que é aplicada às tabelas do banco de dados, não permitindo que valores nulos sejam atribuídos à chave primária de qualquer tabela. Dessa forma, todos os elementos armazenados no banco de dados sempre poderão ser identificados.

2.7 Integridade referencial

Outro aspecto fundamental de integridade do modelo relacional diz respeito aos valores que os atributos que formam a chave estrangeira podem conter. O papel da chave estrangeira é relacionar as linhas de uma tabela com outra. Portanto, para que essa relação possa acontecer, os valores da chave estrangeira devem ser valores que estão presentes na tabela onde aqueles atributos são a chave primária. Caso contrário, a chave estrangeira estaria fazendo referência a elementos inexistentes no banco de dados. Entretanto, é possível que o valor da chave estrangeira seja nulo em uma determinada linha, caso não exista relação daquela linha com qualquer elemento da outra tabela.

2.8 Recomendações para complementação de estudos referentes aos temas deste capítulo:

Recomenda-se a leitura dos capítulos sobre Modelo Relacional das seguintes referências:

CONNOLLY, Thomas M.; BEGG, Carolyn E. **Database systems: a practical approach to design, implementation, and management**. Harlow: Addison-Wesley, 2010.

ELMASRI, Ramez; NAVATHE, Shamkant B. **Sistemas de Banco de Dados**. São Paulo: Addison-Wesley, 2005.

SILBERSCHATZ, Abraham; KORTH, Henry F. **Sistema de banco de dados**. Tradução: Daniel Vieira. 5. ed. Rio de Janeiro: Elsevier, 2006.

2.9 Resumo do capítulo

Neste capítulo foram apresentadas as principais características do modelo relacional. É importante que tenham sido assimilados os seguintes conceitos, que serão necessários para a compreensão dos conteúdos apresentados nos próximos capítulos:

- Modelo relacional
- Relação/Tabela

- Atributo/Coluna
- Chave primária e estrangeira
- Valor nulo
- Integridade de entidade
- Integridade referencial


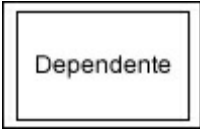
CAPÍTULO 3

MODELAGEM ENTIDADE-RELACIONAMENTO

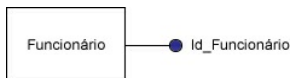
Este capítulo apresenta os conceitos do modelo Entidade-Relacionamento (ER), que é um modelo de dados conceitual de alto nível e muito difundido. O modelo ER é comumente aplicado na fase de elaboração do projeto conceitual de banco de dados e várias ferramentas (*softwares*) de modelagem aplicam seus conceitos.

3.1 Principais conceitos

Os principais conceitos do modelo ER são: entidade, relacionamento, atributo, generalização/especialização e entidade associativa (ou agregação). A cada um desses conceitos corresponde uma notação diagramática no DER conforme apresentado na Figura 13.

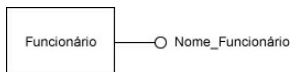
Conceito	Representação/Exemplo	Descrição
Entidade		Uma entidade é no DER como um retângulo que contém um nome da entidade. O nome da entidade deve ser relacionado com o relacionamento.
Entidade Fraca		Uma entidade fraca é representada no DER por um retângulo duplo, onde o texto no interior representa o nome da entidade.

Atributo Identificador



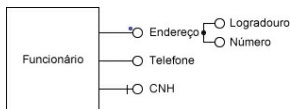
Um atributo ider representado po pintado e ligado por um traço. O identificador é a características c a entidade aond

Atributo (não identificador)



Um atributo (nã é representado vazado e ligado por um traço.

Atributos Compostos, Multivalorados ou Opcionais



Um atributo Cor descrito no diag hierarquia de cir à entidade, com Endereço; um a Multivalorado é i juntamente com cardinalidade, é Telefone; e um a opcional tem a l liga à entidade c um traço, é o ex CNH.

Relacionamento



Um relacioname representado po losango, que co com o nome do relacionamento.

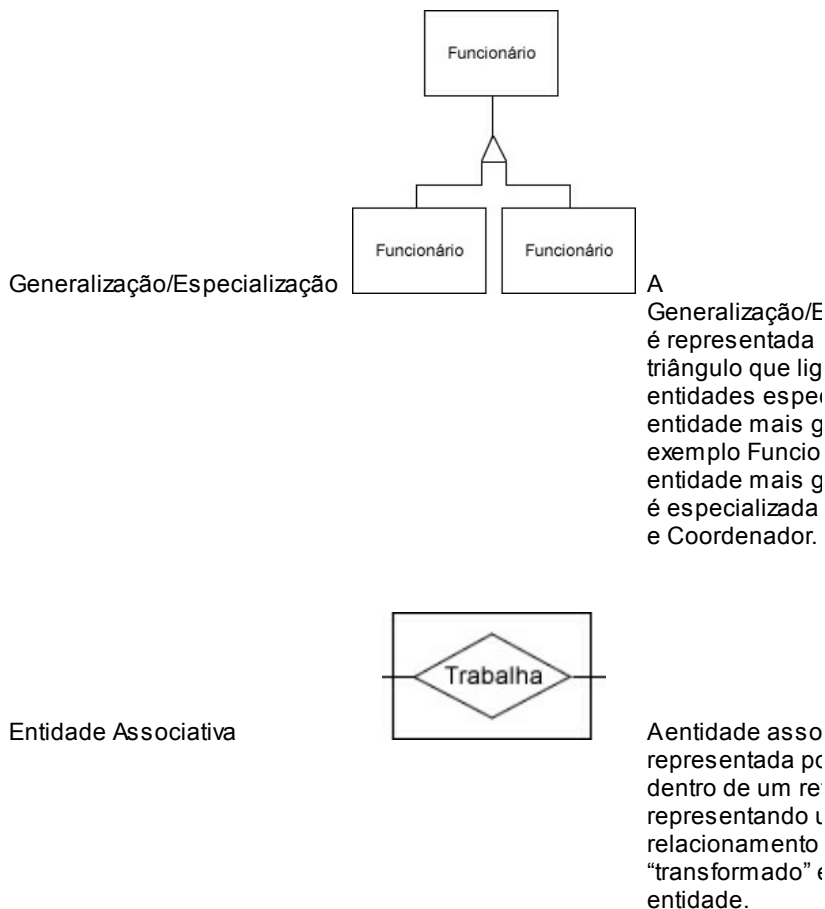


Figura 13 – Conceitos do modelo ER e sua representação diagramática.⁵
 Fonte: elaborada pelo autor.

A Figura 13 apresenta, separadamente, cada um dos principais conceitos do modelo ER, entretanto, o papel da fase de modelagem é criar um modelo único que represente as informações que farão parte do banco de dados, os relacionamentos entre elas e as restrições que se aplicam.

Essas restrições são estabelecidas por elementos como a definição dos atributos identificadores, entre outros. Um elemento muito importante na definição de restrições é a cardinalidade dos relacionamentos, conforme será visto adiante.

3.2 Relacionamentos

Os relacionamentos podem ser descritos em relação ao seu grau, ao número de entidades que relaciona e à sua cardinalidade. A cardinalidade denota quantas ocorrências de uma entidade podem estar associadas a uma determinada ocorrência de outra entidade por aquele relacionamento.

Em relação ao grau, os relacionamentos podem ser: binários, envolvem duas entidades; ternários, envolvem três entidades; quaternários, envolvem quatro entidades; e assim sucessivamente. Na prática, a maior parte dos modelos contempla apenas relacionamentos binários e ternários. Alguns autores inclusive argumentam que qualquer banco de dados poderia ser modelado apenas com relacionamentos binários.

Na notação apresentada neste capítulo, a cardinalidade é expressa em termos de cardinalidade mínima e máxima. A cardinalidade mínima denota o número mínimo de ocorrências de entidade que são associadas a uma ocorrência de uma entidade por meio de um relacionamento e pode ter os valores 0 (zero) ou 1 (um). Já a cardinalidade máxima⁶ denota o número máximo de ocorrências de entidade que são associadas a uma ocorrência de uma entidade por meio do mesmo relacionamento e pode assumir os valores 1 (um) ou N (muitos ou vários). Algumas notações explicitam o número máximo de ocorrências com algum valor específico, quando maior do que 1 (um). Na notação apresentada neste livro é indicado apenas o valor N, quando a cardinalidade máxima é maior do que 1 (um).

3.3 Relacionamento binário

A Figura 14 apresenta os tipos de relacionamentos binários e os classifica em relação ao seu grau e cardinalidade. São apresentados exemplos com a identificação das cardinalidades mínima e máxima.

Cabe observar que alguns DERs apresentados neste capítulo estão sem os atributos (tanto nas entidades como nos relacionamentos). Isso é porque os nomes de tais entidades e relacionamentos são autoexplicativos do que a entidade (ou relacionamento) representa e foram omitidos exclusivamente para efeitos de diagramação do texto deste livro, gerando um texto menos “poluído” e, conseqüentemente, mais claro. Na realidade, para que a técnica de diagramação seja corretamente aplicada na prática, todos os atributos necessários, sejam de entidades ou de relacionamentos, devem ser apresentados no DER.

Conforme visto no capítulo 1, a fase de modelagem conceitual é muito importante no ciclo de vida de projeto de um banco de dados. É nessa fase que os requisitos da situação (aplicação) para a qual se está construindo o banco de dados são explicitados. O modelo Entidade-Relacionamento (ER) é um modelo de dados conceitual e, por ser um modelo de alto nível, permite explicitar esses requisitos, captando ao máximo as características da situação real, sem entrar em detalhes de implementação e específicos do SGBD onde aquele banco de dados será criado. Normalmente, um modelo ER é representado de forma diagramática, por uma técnica chamada Diagrama Entidade-Relacionamento (DER). Existem diferentes notações diagramáticas para a representação de um modelo ER, neste livro, será adotada a técnica chamada Diagrama Entidade-Relacionamento (DER), conforme o modelo originalmente introduzido por Peter Chen, em 1976 (CHEN, 1976).



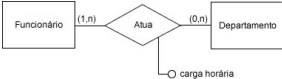
Grau	Cardinalidade	Representação/Exemplo (com cardinalidade mínima e máxima)	Descrição
Binário	1:1		Funcionário gerencia Departamento ⁷
Binário	1:N		Funcionário trabalha em Departamento ⁸
Binário	N:N		Funcionário atua em

Figura 14 – Cardinalidade, grau e exemplos de relacionamentos binários.

Fonte: elaborada pelo autor. Funcionário atua em Projeto.

A cardinalidade mínima também pode ser interpretada como uma restrição de obrigatoriedade ou opcionalidade da participação de ocorrências de uma entidade em um relacionamento. Quando a cardinalidade mínima é zero, está sendo dito que não é obrigatória a participação de uma ocorrência da entidade (a denotada no lado oposto) daquele relacionamento. No exemplo da Figura 14, no relacionamento Gerencia, quando está dito que 1 (um) Funcionário se relaciona com, no mínimo, 1 (um) Departamento, isso significa que não é obrigatório que todas as ocorrências de Funcionário tenham participação nesse relacionamento, ou seja, alguns funcionários não gerenciam departamentos e, conseqüentemente, não participam desse relacionamento.

Alguns relacionamentos também podem conter atributos. Um atributo em um relacionamento é uma informação que não deve estar associada a nenhuma das entidades envolvidas, porque faz sentido somente quando associada ao relacionamento entre elas. É o caso do relacionamento Atua, que aparece na Figura 14. A carga horária de um funcionário em um projeto é uma informação do relacionamento entre essas duas entidades, portanto, do relacionamento Atua.

3.4 Leitura da cardinalidade de relacionamentos binários

Veja, na Figura 15, um trecho de um DER e como deve ser feita a interpretação das cardinalidades do relacionamento.

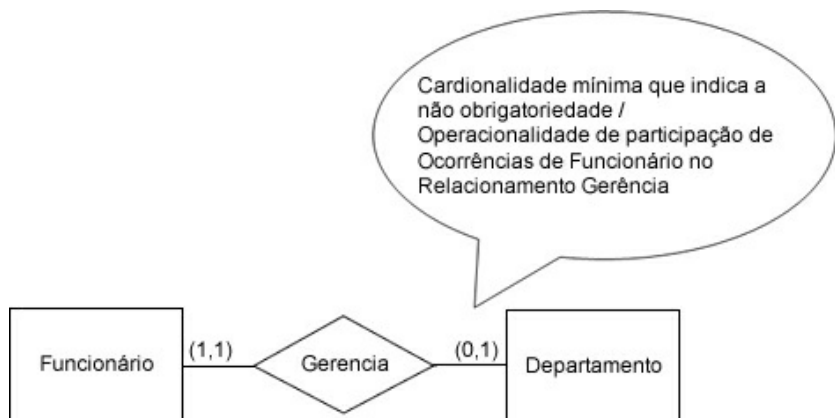


Figura 15 – Relacionamento Gerência – representação diagramática.

Fonte: elaborado pelo autor.

O mesmo relacionamento expresso no DER acima, está descrito, abaixo, com uma notação textual.

Funcionário (1,1) <Gerência> (0,1) Departamento

Figura 16 – Relacionamento Gerência – representação textual com cardinalidades mínima e máxima.

Fonte: elaborado pelo autor.

O relacionamento, representado de uma forma textual (Figura 16), está descrito COM o detalhamento das cardinalidades mínimas (valor do lado esquerdo dentro dos parênteses, o valor verde e o valor vermelho) e com o detalhamento das cardinalidades máximas (valor do lado direito dentro dos parênteses, o valor amarelo e o valor azul).

Conforme visto neste capítulo, a leitura do relacionamento acima pode ser feita das seguintes formas:

- Lendo da esquerda para a direita: 1 (um) funcionário gerencia, **no mínimo, nenhum departamento** e, **no máximo, 1 (um) departamento**.
- Lendo da direita para a esquerda: 1 (um) departamento é

gerenciado por, **no mínimo, 1 (um) funcionário** e, **no máximo, 1 (um) funcionário**.

As cores estão sendo usadas para que se possa identificar onde está, na representação textual do diagrama (Figura 16), a informação sobre a respectiva cardinalidade que aparece no texto de leitura das cardinalidades.

É possível descrever um relacionamento apenas com as cardinalidades máximas. Nesse caso, a representação textual seria a seguinte:

Funcionário **1** <Gerencia> **1** Departamento

Figura 17 – Relacionamento Gerencia – representação textual somente com cardinalidade máxima.

Fonte: elaborado pelo autor.

O relacionamento acima (Figura 17), representado de uma forma textual, está descrito SEM o detalhamento das cardinalidades mínimas e somente COM o detalhamento das cardinalidades máximas.

A leitura do relacionamento acima é feita da seguinte forma:

- Lendo da esquerda para a direita: 1 (um) funcionário gerencia, **no máximo, 1 (um) departamento**.
- Lendo da direita para esquerda: 1 (um) departamento é gerenciado por, **no máximo, 1 (um) funcionário**.

A cardinalidade mínima oferece mais riqueza ao modelo. Conforme explicado acima, a cardinalidade mínima também pode ser interpretada como uma restrição de obrigatoriedade ou opcionalidade da participação de ocorrências de uma entidade em um relacionamento. Quando a cardinalidade mínima é zero, está sendo dito que não é obrigatória a participação de uma ocorrência da entidade (a entidade do lado oposto) naquele relacionamento. No exemplo, no relacionamento Gerencia, quando está dito que **1 (um) Funcionário** se relaciona com, no mínimo, **0 Departamento**, isso significa que não é obrigatório que todas as ocorrências de Funcionário tenham participação no relacionamento, ou seja, alguns funcionários não gerenciam departamentos e, conseqüentemente, não participam do relacionamento.

Assim, é possível ter os valores 0 ou 1 para cardinalidade mínima, o 0 representando que a entidade não participa necessariamente do relacionamento e o 1 que ela participa obrigatoriamente, pelo menos uma ocorrência dela.

3.5 Relacionamento ternário

A Figura 18 mostra o exemplo de um relacionamento ternário. A cardinalidade em um relacionamento ternário refere-se a pares de entidades. No relacionamento Distribuição, a cardinalidade 1 indica que um par Cidade, Produto se relaciona com 1 Distribuidor. O relacionamento deve ser lido dessa mesma forma para cada par de entidades.

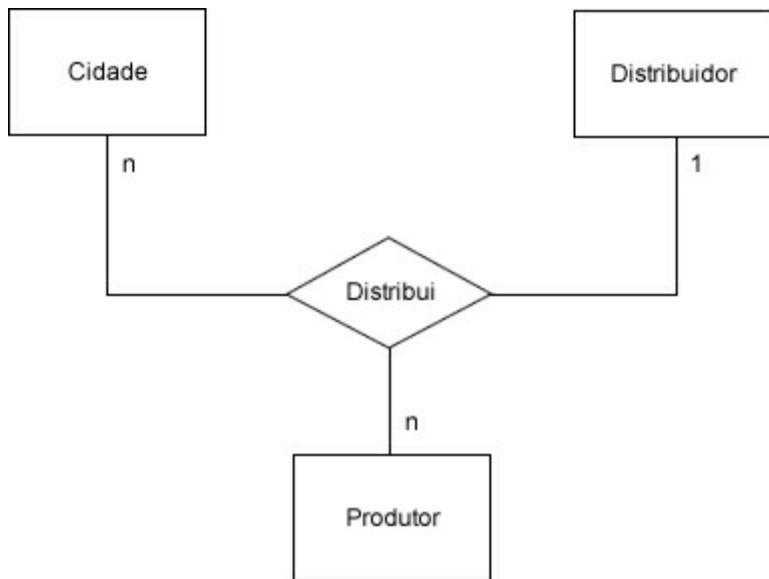


Figura 18 – Exemplo de relacionamento ternário.

Fonte: elaborado pelo autor.

3.6 Relacionamento reflexivo

O Relacionamento reflexivo, ou autorrelacionamento, é quando uma entidade relaciona-se com ela mesma. Por exemplo, o relacionamento entre funcionários que exercem o papel de líderes uns dos outros é um relacionamento da entidade Funcionário com ela mesma. Veja o exemplo na Figura 19.

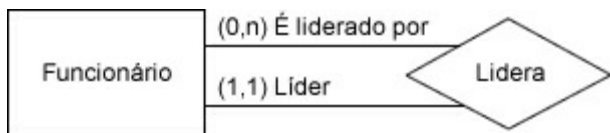


Figura 19 – Exemplo de relacionamento reflexivo (autorrelacionamento).

Fonte: elaborado pelo autor.

No DER da Figura 19, as inscrições “É liderado por” e “Lider” são os papéis que a entidade Funcionário exerce ao participar do relacionamento Lidera. Lendo o DER, iniciando pela linha de baixo: 1 (um) funcionário é líder de, no mínimo, nenhum outro funcionário e, no máximo, N funcionários; lendo o DER, iniciando pela linha de cima: 1 (um) funcionário é liderado por, no mínimo, 1 (um) outro funcionário e, no máximo, 1 (um) outro funcionário.

3.7 Entidade associativa

Entidade é o conceito usado para representar um elemento da realidade que está sendo analisada e que deve constar no modelo. Entidade associativa é o conceito que transforma um relacionamento em uma entidade para que possa ser ligado, diretamente, a outros relacionamentos com entidade do modelo.

Veja, por exemplo, o seguinte trecho da descrição de uma realidade a ser modelada: “Uma empresa que dá treinamento na área de informática tem a seguinte programação semestral de cursos...”

Ao analisar a descrição, é possível identificar que curso tem informações importantes e que deve, portanto, fazer parte do banco de dados, logo, CURSO é identificado como uma entidade no Diagrama ER (Figura 20).

Na descrição da situação, também aparece a informação de que os cursos são ofertados em determinados períodos (“...programação

semestral de cursos..."). Assim, **PERÍODO** também é definido como entidade (Figura 20), e deve ser ligado ao **CURSO** pelo relacionamento **OFERTADO** (Figura 20).

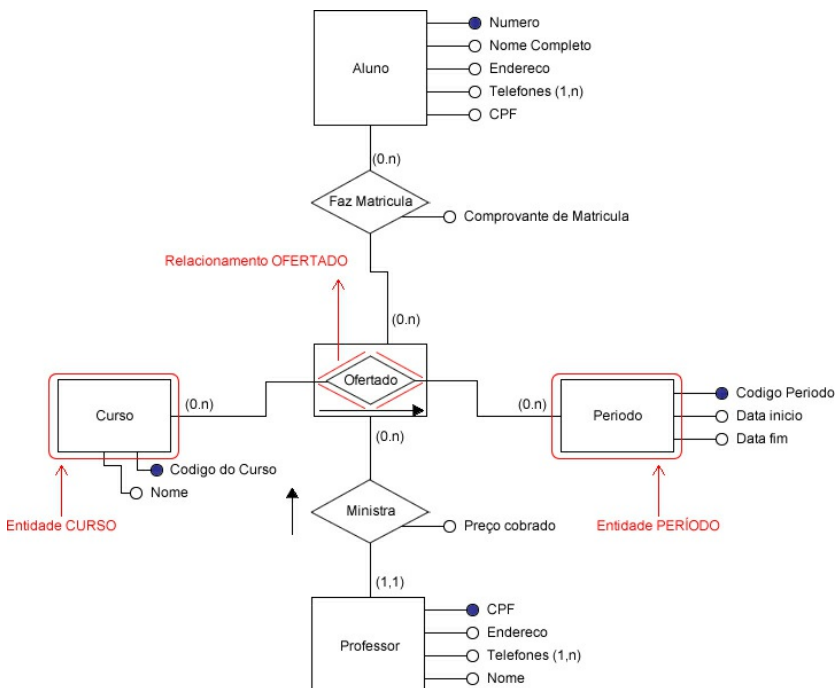


Figura 20 – DER que representa a oferta de turmas de cursos em determinados períodos.

Fonte: elaborado pelo autor.

É possível identificar que as ofertas de cursos em determinados períodos são, de fato, as turmas, as quais são ministradas por professores e onde os alunos fazem matrícula. Mas a entidade **TURMA** não está no diagrama. Por isso, o **Relacionamento N:N OFERTADO** é transformado em uma **Entidade Associativa**, para que seja possível, então, estabelecer o relacionamento dele com outras entidades, já que não é possível estabelecer relacionamento entre um relacionamento e uma entidade, somente é possível criar relacionamentos entre entidades.

Assim, na Figura 21, abaixo, apresenta a aplicação do conceito de **Entidade Associativa** para transformar o **relacionamento N:N OFERTADO** na entidade **TURMA**.

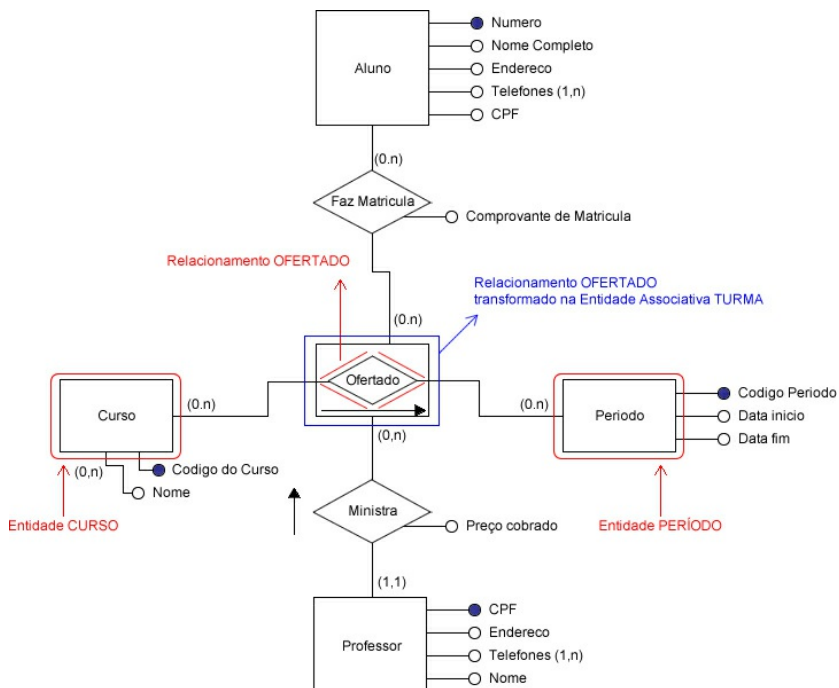


Figura 21 – Exemplo de aplicação do conceito de entidade associativa.

Fonte: elaborado pelo autor.

É importante observar que a única situação onde uma Entidade Associativa é necessária é quando se deseja transformar um relacionamento em uma entidade, conforme explicado acima. E, caso o relacionamento que foi transformado na Entidade Associativa tenha atributos, a Entidade Associativa passa a ter exatamente os mesmos atributos, sendo que não é necessário representá-los novamente. Por outro lado, uma Entidade Associativa não deve ser usada apenas que o relacionamento tenha atributos, um relacionamento pode ter atributos sendo apenas um relacionamento, conforme é apresentado na Figura 14,

no relacionamento Funcionário atua em Projeto.

3.8 Um exemplo completo

A seguir, é apresentado um exemplo completo de um DER sobre a situação de uma Universidade, com seus cursos, disciplinas, turmas, professores, alunos matriculados e seus históricos escolares. Em breve o DER apresentado será detalhado.

Como uma forma padronizada para a leitura de DER, pode adotar o seguinte critério: ler o diagrama da esquerda para a direita e de cima para baixo. Esse mesmo critério pode ser adotado para a elaboração de um DER e decisão de que nomes dar aos relacionamentos, já que se lidos em um sentido podem ter um nome e no sentido contrário outro. Por exemplo, o relacionamento “Leciona” poderia chamar-se “É lecionada por”, caso a leitura do diagrama for feita a partir da entidade turma. Além disso, quando um DER é mais extenso, é possível colocar as cardinalidades mais próximas da representação dos relacionamentos para facilitar a compreensão do leitor.

Segundo o DER da Figura 22, é possível verificar que a Universidade admite alunos estrangeiros ou locais, caso sejam estrangeiros, as informações guardadas sobre eles são a sua nacionalidade e o número de um documento de identidade, atributos da entidade Estrangeiro. Caso sejam locais, o número do CPF é documento que será armazenado conforme está indicado na entidade Local. Entretanto, como é usada uma Generalização/Especialização, a entidade Aluno registra os atributos que pertencem tanto aos alunos estrangeiros quanto locais, que são número de Matrícula e Nome.

A entidade Departamento representa os departamentos da Universidade e permite que sejam guardadas as seguintes informações sobre eles: código, nome, sala e telefone. Os departamentos contêm cursos, que estão representados pela entidade Curso. Sobre cada curso poderá ser guardado o seu número, carga horária total, nome e uma breve descrição.

As disciplinas estão representadas pela entidade Disciplina, que participa de vários relacionamentos, o que é natural já que é um elemento central na organização de uma Universidade. Uma disciplina tem código e nome. As disciplinas pertencem a cursos. E, note, o relacionamento Pertence tem o atributo Semestre, pois cada disciplina pode estar em semestres diferentes para cursos distintos. Disciplina também tem

relacionamentos com Professor, Turma e Aluno. Esses relacionamentos serão lidos no sentido em que foi indicado no início dessa seção.

Professor é responsável por disciplinas. Sobre o professor são guardados o número de matrícula, nome e seus telefones e ele pode lecionar em várias turmas. O atributo telefone foi modelado como um atributo multivalorado.

A entidade Turma permite que sejam armazenados o número e a sala onde ocorrem as aulas. E uma turma é de uma determinada disciplina e permite a matrícula de vários alunos.

Por fim, o aluno cursa disciplinas e sobre cada uma das disciplinas cursadas ficam registrados o semestre e a nota que ele recebeu.

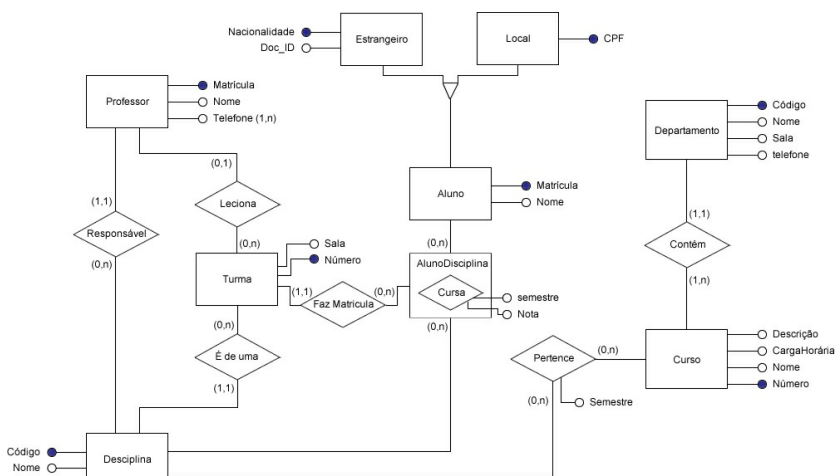


Figura 22 – DER de uma Universidade.

Fonte: elaborado pelo autor.

3.9 Construindo o DER a partir da descrição textual de uma situação

A construção de um DER dá-se a partir do contato do profissional que está responsável por levantar os requisitos do sistema com o cliente que encomenda tal sistema. No projeto do DER, também podem ser usados como subsídio documentos relativos à situação da empresa a ser modelada, assim como descrições textuais sobre a situação. Uma

sugestão prática de como derivar entidades, atributos e relacionamentos a partir de uma situação descrita de forma textual é identificar entidades e atributos a partir dos substantivos presentes no texto e os relacionamentos a partir dos verbos que conectam tais substantivos nas frases do texto.

Abaixo, é apresentado um exemplo, onde o DER é criado a partir da descrição textual de uma dada situação, que é a seguinte:

“Uma empresa, que dá treinamento na área de informática, tem a seguinte programação semestral de **cur**so: DBA para o Banco de Dados Oracle, Introdução à Java, Java Avançado e outros. Um curso pode ser **ministrado** (**Ofertado – é a turma**) várias vezes no semestre. Por exemplo, o curso de DBA Oracle para o primeiro semestre de 2013 teve a seguinte programação: a primeira turma de 15/03/13 a 30/03/13 e a segunda turma de 20/06/13 a 05/07/13. Já o curso de Introdução à Java foi programado para: a primeira turma no mesmo período da primeira turma de DBA Oracle e a segunda turma para o **período** de 10/08/13 a 30/08/13. Para fazer a matrícula, o **aluno** precisa apresentar: **CPF**, **nome completo**, **endereço**, **telefones de contato** e o **comprovante do pagamento da taxa de matrícula**, o **curso que pretende fazer e em qual período** (Faz matrícula). No momento da matrícula, a cada aluno é atribuído **um número que o identifica** entre todos os alunos da turma. Se o aluno, eventualmente, volta a cursar mais cursos, o número dele continua o mesmo. Há que se **manter o registro** (**Tem**) para cada **aula**, de um **resumo sucinto da matéria lecionada**, da **data** e da **frequência de cada aluno** (Frequenta). Para cada turma de cada curso é **alocado** (**Ministra**) um **professor** cujos dados de interesse são: **CPF**, **nome**, **endereço**, **preço cobrado para ministrar o curso**, **telefones para contato** e **quais os cursos que o professor pode ministrar** (Pode ministrar).

No texto acima, estão em **vermelho** os substantivos *indicativos* de entidade, em **azul** de atributos e em **verde** de relacionamentos ou atributos de relacionamento.

Apartir disso, o DER, que representa tal situação, é apresentado na Figura 23.

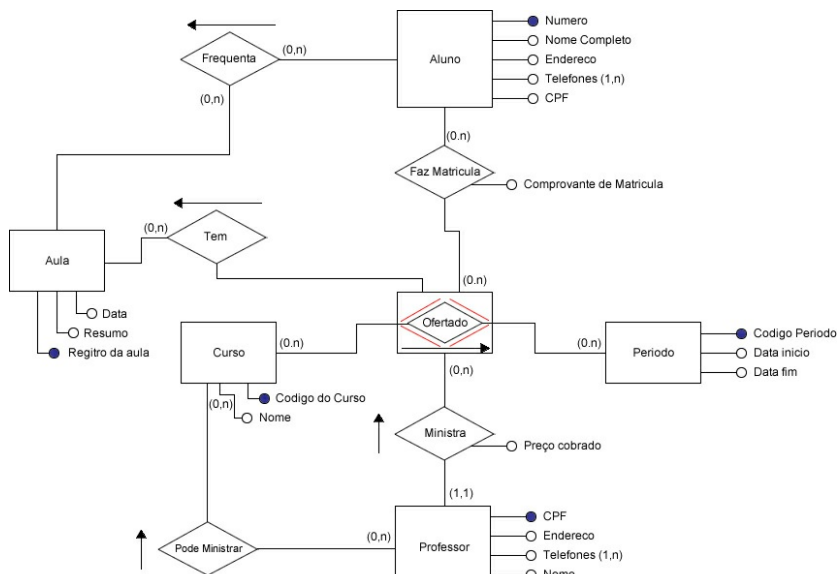


Figura 23 – DER derivado a partir da análise de uma descrição textual.
Fonte: elaborado pelo autor.

No DER da Figura 23, as setas representam o sentido de leitura dos relacionamentos, isso tem o objetivo de facilitar a leitura e interpretação do DER como um todo. Por exemplo, a seta da direita para a esquerda, logo acima do relacionamento FREQUENTA induz à leitura desse relacionamento da seguinte forma: “Aluno frequenta Aulas”; assim como a seta de baixo para cima ao lado do relacionamento PODE MINISTRAR induz à leitura desse relacionamento como: “Professor PODE MINISTRAR Curso”.

3.10 Recomendações para complementação de estudos referentes aos temas deste capítulo

Algumas bibliografias apresentam notações alternativas para a descrição do modelo conceitual. Algumas dessas notações são apresentadas em:

CONNOLLY, Thomas M.; BEGG, Carolyn E. Database systems: a practical approach to design, implementation, and management. Harlow: Addison-Wesley, 2010.

ELMASRI, Ramez; NAVATHE, Shamkant B. Sistemas de Banco de Dados. São Paulo: Addison-Wesley, 2005.

3.11 Resumo do capítulo

Este capítulo apresentou os conceitos do Modelo Entidade-Relacionamento (ER), que é um modelo de dados conceitual de alto nível. Foram abordados os conceitos que permitem a elaboração do projeto conceitual de um banco de dados. É importante que esses conceitos sejam bem assimilados, pois o próximo capítulo irá apresentar as regras de mapeamento do modelo conceitual para o modelo lógico.

⁵ Os Diagramas ER apresentados neste livro foram desenhados usando a ferramenta BrModelo. Essa ferramenta foi desenvolvida por Carlos Henrique Cândido sob a orientação do Prof. Dr. Ronaldo dos Santos Mello (UFSC), como trabalho de conclusão do curso de pós-graduação em banco de dados (UNVAG - MT e UFSC). O BrModelo é um software livre e está disponível para download em <http://www.sis4.com/brModelo/>. A notação empregada pela ferramenta é baseada na apresentada no livro Projeto de Banco de Dados, do professor Carlos Alberto Heuser (UFRGS) (HEUSER, 2004).

⁶ A indicação da cardinalidade mínima e máxima no modelo conceitual (modelo ER/DER) tem o objetivo de agregar mais informação ao modelo e indicar possibilidades para a fase de implementação, definição dos modelos lógico e físico. Por exemplo, indicar cardinalidade mínima 0 pode determinar como o programa que vai realizar a inserção de dados no banco de dados vai ser codificado.

⁷ A leitura deste diagrama pode ser feita das seguintes formas:

- da esquerda para a direita: 1 funcionário gerencia, no mínimo, nenhum departamento e, no máximo, 1 departamento.
- da direita para a esquerda: 1 departamento é gerenciado por, no mínimo, 1 funcionário e, no máximo, 1 funcionário.

⁸ A leitura deste diagrama pode ser feita das seguintes formas:

- 1 funcionário trabalha em, no mínimo, 1 departamento e, no máximo, 1 departamento.
- 1 departamento tem, no mínimo, nenhum funcionário e, no máximo, N

funcionários trabalhando nele.

9 A leitura deste diagrama pode ser feita das seguintes formas:

- 1 funcionário atua em, no mínimo, nenhum projeto e, no máximo, N projetos.
- 1 projeto tem, no mínimo, 1 funcionário e, no máximo, N funcionários atuando nele.

CAPÍTULO 4

MAPEAMENTO DE MODELO ENTIDADE-RELACIONAMENTO PARA MODELO RELACIONAL

O modelo ER é mais abstrato do que o modelo relacional e, portanto, contém uma série de informações que não podem ser representadas diretamente no modelo relacional. Nesse sentido, é necessário que seja feito um mapeamento de um modelo para o outro, de forma que não se perca a informação descrita no nível mais abstrato. Este capítulo apresenta as regras de mapeamento de um modelo Entidade-Relacionamento para o modelo relacional.

Conforme visto no capítulo 1, o projeto lógico de um banco de dados é o processo de construção de um modelo, baseado em um modelo de dados (relacional, por exemplo), mas independente de um SGBD específico e de outras considerações físicas. O ponto de partida para a construção do projeto lógico é o projeto conceitual, que foi trabalhado no capítulo 3. A seguir serão apresentadas as regras de mapeamento entre os dois modelos, considerando o que foi apresentado nos capítulos 2 e 3, inclusive a notação adotada. As regras de mapeamento apresentadas têm por objetivo criar um projeto lógico de banco de dados que permita às aplicações realizarem suas operações sobre ele com mais eficiência. As ideias apresentadas neste capítulo estão baseadas no trabalho apresentado em (HEUSER, 2004).

4.1 Mapeamento de entidades

Em geral, as entidades são mapeadas para relações no modelo relacional. Sendo o nome da relação o mesmo nome da entidade no modelo conceitual e os atributos da relação os atributos da entidade. O atributo identificador da entidade será o atributo chave da relação. As entidades fracas são mapeadas da mesma forma, criando-se um atributo como chave estrangeira e com a restrição de não poder receber o valor nulo, permitindo, assim, estabelecer-se o relacionamento com a entidade forte à qual ela é vinculada.

4.2 Mapeamento de atributos

Os atributos passam a ser os atributos da relação. Atributos identificadores são as chaves da relação.

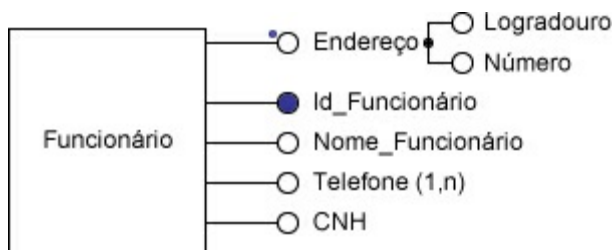
Em relação aos atributos compostos, deve ser tomada uma decisão de projeto e existem duas possibilidades: (a) cada um dos atributos de nível mais baixo na hierarquia definida é mapeado como atributo da relação; (b) cria-se um único atributo, o mais genérico da hierarquia. As duas alternativas têm vantagens e desvantagens. Na opção (a), a relação tem mais atributos para serem gerenciados, entretanto a informação estará mais detalhada, permitindo o registro e acesso mais elaborados. Na opção (b), a informação fica em apenas um atributo, o modelo fica mais simples, mas não há os detalhes da opção anterior.

Atributos Multivalorados são mapeados com a criação de uma nova entidade e o relacionamento dela com a entidade onde o atributo foi originalmente criado.

E, por fim, atributos opcionais são mapeados para atributos da relação e é indicada explicitamente uma restrição de que esse atributo pode conter o valor nulo.

A Figura 26 apresenta as regras de mapeamento de atributos com exemplos.

Modelo ER



Modelo relacional

Relação Funcionário	
Atributos	Restrições
Id_Funcionário	Chave Primária

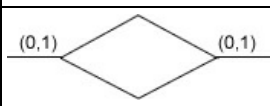
Nome_Funcionário	Não permite valor nulo
CNH	Permite valor nulo
Logradouro	Não permite valor nulo
Número	Não permite valor nulo
Relação Telefone	
Atributos	Restrições
Id_Funcionário	Chave Estrangeira, que relaciona Id_Funcionário com a Relação Funcionário
Id_Telefone	Chave Primária
Número	Não permite valor nulo

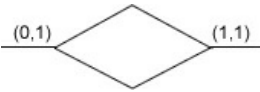
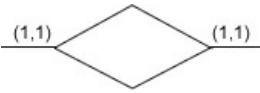
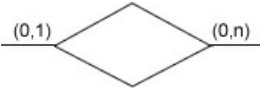
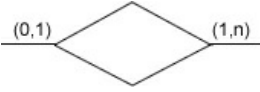
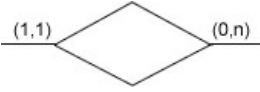
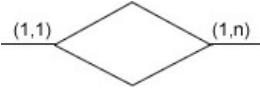

Figura 26 – Regras de mapeamento de atributos com exemplos.

Fonte: elaborado pelo autor.

4.3 Mapeamento de relacionamentos binários

Os relacionamentos podem tanto ser mapeados para atributos como para relações. A definição dar-se-á em função da cardinalidade do relacionamento. Segundo Heuser (2004) apresenta um conjunto de regras que, levando em consideração os valores de cardinalidade mínima e máxima dos relacionamentos, são sugeridas alternativas de mapeamento. As alternativas se diferenciam em relação ao número de relações e atributos que vão resultar no modelo gerado. A Figura 18 apresenta esse conjunto de regras de forma resumida.

Relacionamentos 1:1			
Tipo de relacionamento	Regra de implementação		
	Tabela própria	Adição de coluna	Fusão de tabelas
	Pode ser usada	Alternativa preferida	Não usar

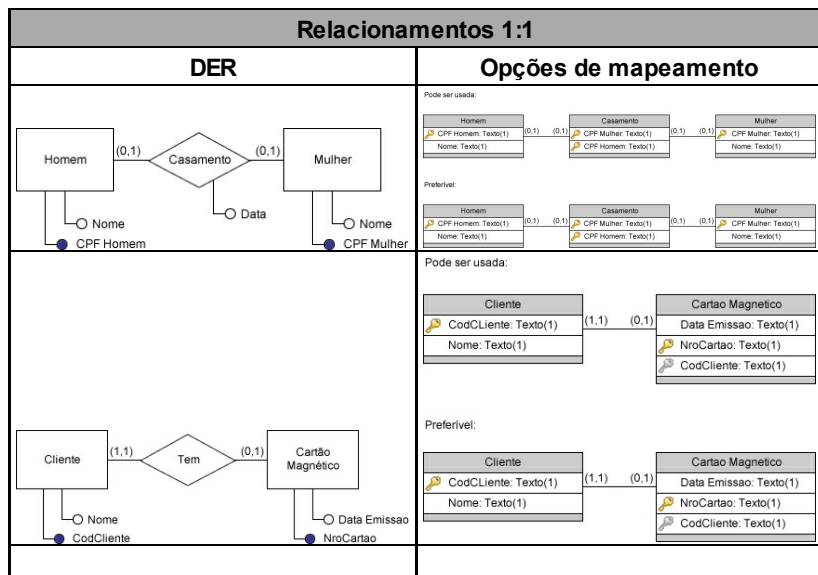
	Não usar	Pode ser usada	Alternativa preferida
	Não usar	Não usar	Alternativa preferida
Relacionamentos 1:N			
Tipo de relacionamento	Regra de implementação		
	Tabela própria	Adição de coluna	Fusão de tabelas
	Pode ser usada	Alternativa preferida	Não usar
	Pode ser usada	Alternativa preferida	Não usar
	Não usar	Alternativa preferida	Não usar
	Não usar	Alternativa preferida	Não usar
Relacionamentos N:N			
Tipo de relacionamento	Regra de implementação		
	Tabela própria	Adição de coluna	Fusão de tabelas
	Alternativa	Não usar	Não usar

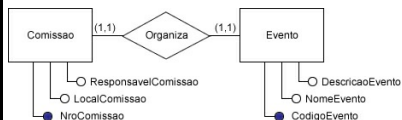
	preferida		
	Alternativa preferida	Não usar	Não usar
	Alternativa preferida	Não usar	Não usar

Figura 27 – Regras para mapeamento de relacionamentos.

Fonte: HEUSER, 2004.

Para cada um dos casos acima, a Figura 28 apresenta exemplos, considerando o mapeamento pelas alternativas possíveis e preferida, identificadas na Figura 27. Os DERs foram construídos com a ferramenta brModelo, e o mapeamento gerado também pela ferramenta, com os ajustes necessários.



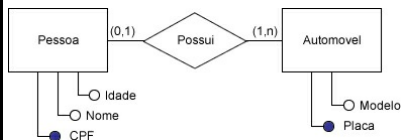


Preferível:

Comissao+Evento	
NroComissao:	Texto(1)
ResponsavelComissao:	Texto(1)
LocalComissao:	Texto(1)
CodigoEvento:	Texto(1)
NomeEvento:	Texto(1)
DescricaoEvento	Texto(1)

Relacionamentos 1:N

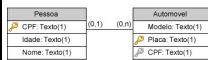
DER



Obs.: como para o caso de um relacionamento 1:N, com parcialidade do lado 1 e obrigatoriedade do lado N, o mesmo mapeamento é sugerido na Figura 18, o exemplo não será apresentado nesta Figura.

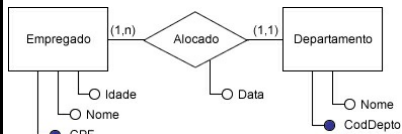
Opções de mapeamento

Pode ser usado:



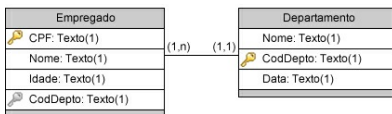
Preferível:



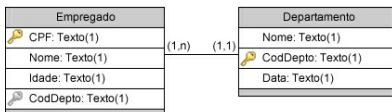


Obs.: como para o caso de um relacionamento 1:N, com obrigatoriedade do lado 1 e parcialidade do lado N, o mesmo mapeamento é sugerido na Figura 18, o exemplo não será apresentado nesta Figura.

Pode ser usada:

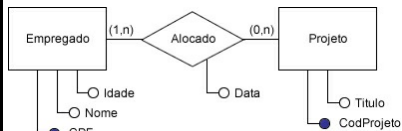


Preferível:



Relacionamentos N:N

DER



Obs.: como para o caso de todas as opções de cardinalidade (mínima) em relacionamentos binários N:N sempre o mesmo mapeamento é sugerido na Figura 18, os demais casos exemplo não serão apresentados nesta Figura.

Opções de mapeamento

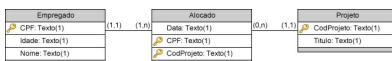


Figura 28 – Exemplo de aplicação das regras para mapeamento de relacionamentos apresentadas na Figura 27.

Fonte: elaborado pelo autor.

4.4 Relacionamento reflexivo

O mapeamento de relacionamentos reflexivos segue as mesmas regras de qualquer mapeamento binário, assim como o mapeamento de Entidade Associativa em relacionamento binário. Ambos são relacionamentos binários e, por isso, podem seguir as mesmas regras de mapeamento.

4.5 Mapeamento de Generalizações/Especializações

O mapeamento de Generalizações/Especializações pode ser feito de duas formas:

- a. mapeia-se toda a hierarquia em apenas uma relação. A chave primária é o atributo identificador da entidade mais genérica. É necessário que todos os atributos das entidades especializadas sejam definidos com a restrição de que permitem valores nulos, pois todos eles estarão fazendo parte da mesma tabela do banco de dados, mas, como a tabela representa várias entidades especializadas, muitos deles poderão ficar sem valores atribuídos. É preciso acrescentar ao projeto um atributo Tipo para identificar a qual entidade na hierarquia de especialização uma determinada linha da tabela representa. A vantagem dessa alternativa é que todos os dados ficam em uma única tabela. Entretanto, a desvantagem é que existem muitos campos opcionais (aqueles que permitem valores nulos), além do atributo Tipo, que foi criado apenas para identificar a que entidade especializada do modelo conceitual os dados de uma determinada linha se relacionam. A Figura 29 apresenta um exemplo de modelo conceitual (ER) e o respectivo mapeamento para o modelo lógico (relacional).

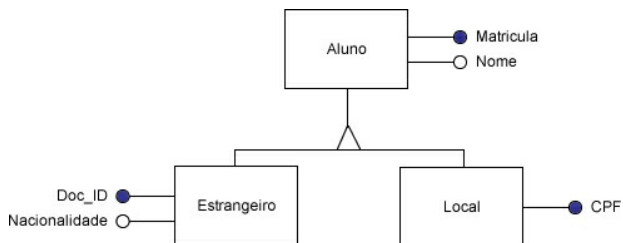


Tabela Aluno	
Atributo	Restrições
Matrícula	Chave Primária
Nome	
Tipo	NOT NULL
Doc_ID	NULL
Nacionalidade	NULL
CPF	NULL

Figura 29 – Exemplo de mapeamento de Especialização/Generalização para relação única.

Fonte: elaborado pelo autor.

- b. mapeia-se cada entidade, tanto a genérica como as especializadas, nas relações independentes, com a chave primária de cada relação sendo o atributo identificador da entidade mais genérica.

A vantagem dessa alternativa é que não existem os atributos opcionais (permitem o valor nulo), serão definidos como opcionais apenas aqueles em que os dados da realidade modelada realmente possam ser nulos. Como desvantagem a chave primária de todas as tabelas que representam completamente a hierarquia é a mesma. A Figura 30 apresenta um exemplo de modelo conceitual (ER) e o respectivo mapeamento para o modelo lógico (relacional).

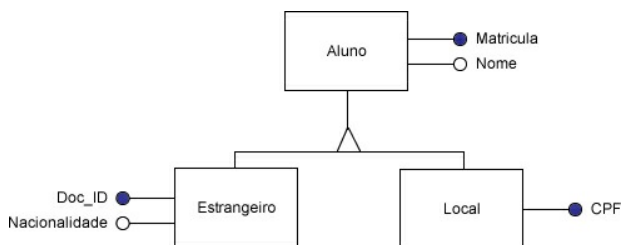


Tabela Aluno	
Atributo	Restrições

Matrícula	Chave Primária
Nome	

Tabela Estrangeiro	
Atributo	Restrições
Matrícula	Chave Primária
Doc_ID	
Nacionalidade	

Tabela Local	
Atributo	Restrições
Matrícula	Chave Primária
CPF	

Figura 30 – Exemplo de mapeamento de Especialização/Generalização para várias relações.
Fonte: elaborado pelo autor.

4.6 Um exemplo completo

A Figura 31 apresenta um trecho do DER já apresentado na Figura 23, do capítulo sobre Modelagem ER. O objetivo é fazer uma discussão sobre como aplicar as regras de mapeamento para este trecho de DER.

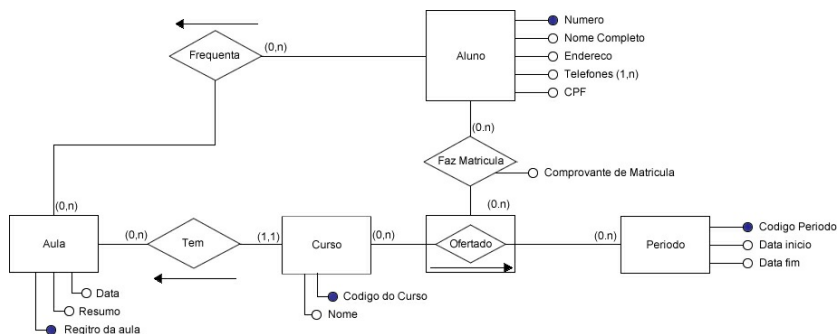


Figura 31 – DER para um exemplo completo de mapeamento.

Fonte: elaborado pelo autor.

A Figura 32 apresenta o conjunto de tabelas resultante do processo de mapeamento do DER da Figura 31 para tabelas de um banco de dados relacional, empregando as regras de mapeamento descritas neste capítulo. Cada tabela é apresentada com seu nome, conjunto de atributos e chaves primárias e estrangeiras, junto com uma breve explicação sobre que elementos do DER da Figura 31 deram origem a ela.

Nome da Tabela: Aluno	
Atributos: Número, Nome_Completo, Endereço, CPF	
Chave primária: Número	
Chave Estrangeira: não tem	Tabela que referencia: não se aplica
A tabela Aluno é gerada a partir do mapeamento da entidade Aluno	

Nome da Tabela: Telefone	
Atributos: NúmeroTelefone, Número Aluno	
Chave primária: NúmeroTelefone	
Chave Estrangeira: NúmeroAluno	Tabela que referencia: Aluno

A tabela Telefone é gerada a partir do mapeamento do atributo multivalorado Telefone da tabela Aluno

Nome da Tabela: Curso
Atributos: CódigoDoCurso, Nome

Chave primária: CódigoDoCurso

Chave Estrangeira: não tem

Tabela que referencia: não se aplica

A tabela Curso é gerada a partir do mapeamento da entidade Curso

Nome da Tabela: Período
Atributos: CódigoPeríodo, DataInício, DataFim

Chave primária: CódigoPeríodo

Chave Estrangeira: não tem

Tabela que referencia: não se aplica

A tabela Período é gerada a partir do mapeamento da entidade Período

Nome da Tabela: Ofertado
Atributos: CódigoCurso, CódigoPeríodo

Chave primária: CódigoCurso, CódigoPeríodo

Chave Estrangeira: CódigoCurso
Chave Estrangeira: CódigoPeríodo

Tabela que referencia: Curso
Tabela que referencia: Período

A tabela Ofertado é gerada a partir do mapeamento do relacionamento binário N:N Ofertado

Nome da Tabela: FazMatrícula

Atributos: CódigoCurso, CódigoPeríodo, NúmeroAluno, ComprovanteDeMatrícula	
Chave primária: CódigoCurso, CódigoPeríodo, NúmeroAluno	
Chave Estrangeira: CódigoCurso, CódigoPeríodo Chave Estrangeira: NúmeroAluno	Tabela que referencia: Ofertado Tabela que referencia: Aluno
A tabela FazMatrícula é gerada a partir do mapeamento do relacionamento binário N:N Faz Matrícula	

Nome da Tabela: Aula Atributos: RegistroDaAula, Data, Resumo, CódigoDoCurso	
Chave primária: RegistroDaAula	
Chave Estrangeira: CódigoDoCurso	Tabela que referencia: Curso
A tabela Aula é gerada a partir do mapeamento da entidade Aula o relacionamento binário 1:N Tem (de Curso Tem Aula)	

Nome da Tabela: Frequenta Atributos: RegistroDaAula, NúmeroAluno	
Chave primária: RegistroDaAula, NúmeroAluno	
Chave Estrangeira: RegistroDaAula Chave Estrangeira: NúmeroAluno	Tabela que referencia: Aula Tabela que referencia: Aluno
A tabela Frequenta é gerada a partir do mapeamento do relacionamento binário N:N Frequenta	

Figura 32 – Mapeamento do DER da Figura 31.

Fonte: elaborado pelo autor.

Com o objetivo de compreender os conceitos sobre mapeamento, compare as informações de cada tabela da Figura 32 com o DER da Figura 31.

4.7 Recomendações para complementação de estudos referentes aos temas deste capítulo

Recomenda-se a leitura do livro:

HEUSER, Carlos Alberto. **Projeto de Banco de Dados**. 5. Ed. Porto Alegre: Sagra Luzzatto, 2004.

4.8 Resumo do capítulo

Neste capítulo foram apresentadas as regras para o mapeamento do projeto conceitual de um banco de dados, modelado a partir da abordagem ER, para o projeto lógico, baseado no modelo relacional. É importante que as regras de mapeamento sejam bem compreendidas e aplicadas corretamente na geração do projeto lógico, possibilitando, assim, o correto funcionamento e bom desempenho do SGBD ao manipular o banco de dados depois de criado.

CAPÍTULO 5

NORMALIZAÇÃO

O projeto conceitual de um banco de dados também pode ser feito pela elaboração direta do conjunto de relações que formam o banco de dados a partir do levantamento dos requisitos da aplicação. Este capítulo apresenta a Normalização, uma técnica para projeto de banco de dados relacionais, que auxilia na identificação do melhor agrupamento dos dados de forma a atender os requisitos da aplicação.

Normalização é uma técnica de projeto que estabelece que os dados sejam analisados em relação às suas dependências funcionais e, a partir da aplicação de algumas regras envolvendo essas dependências, determina o conjunto de tabelas que farão parte do banco de dados. Seguindo a técnica de normalização, espera-se que o conjunto de tabelas resultantes no banco de dados permita o armazenamento de dados com o mínimo de redundância possível e, conseqüentemente, gere condições para que o banco de dados permaneça íntegro e tenha seus processos de manutenção facilitados.

O princípio dessa técnica consiste em analisar o conjunto de dados que deve fazer parte do banco de dados, a partir de relatórios, informações sobre a situação, ou do próprio resultado da fase de levantamento de requisitos, e agrupar esses dados de acordo com a aplicação de algumas regras que serão explicadas mais adiante. Essas regras são denominadas de formas normais.

5.1 Primeira Forma Normal (1FN)

Definição: uma tabela está na primeira forma normal (1FN) se, e somente se, todos os atributos tiverem apenas valores atômicos, ou seja, se cada atributo só puder ter um valor para cada linha da tabela.

Uma tabela é dita não normalizada quando possui um ou mais grupos repetitivos de dados, ou seja, possui tabelas aninhadas uma dentro das outras. Na Figura 21 é apresentado o conjunto de dados não normalizado, que descreve os projetos desenvolvidos em uma empresa e os funcionários que trabalham nesses projetos.

Funcionários do Projeto:

Id_Funcionário	Nome_Funcionário	Categoria	Salário	Data de Início	Car Hor
100	João	1	3.000	10/10/2008	12
200	Pedro	2	3.500	12/05/2008	8
400	Ana	2	3.500	15/07/2008	8

Id_Projeto: 2 Descrição: Implantação de ERP Orçamento: 100 mil

Funcionários do Projeto:

Id_Funcionário	Nome_Funcionário	Categoria	Salário	Data de Início	Car Hor
100	João	1	3.000	13/11/2009	8
300	Maria	1	3.000	10/12/2008	12
500	Luiz	2	3.500	05/10/2009	8

Figura 33 – Exemplo de um conjunto de dados não normalizado.¹⁰

Fonte: adaptada de material didático elaborado originalmente pelo professor Ronaldo dos Santos Mello da UFSC.

O conjunto de dados da Figura 33 não está normalizado, pois existem atributos que não tem domínio atômico, ou seja, existem grupos repetitivos dentro da tabela. É o caso do atributo Funcionário, que tem como domínio uma tabela descrevendo o conjunto de funcionários que trabalham no projeto.

Para que esse conjunto de dados atenda às restrições da primeira forma normal, é preciso eliminar os grupos repetitivos e tornar a tabela plana, ou seja colocar todos os atributos no mesmo nível da tabela principal, definindo como chave primária uma chave composta pelos atributos identificadores de cada grupo identificado. No caso do exemplo apresentado na Figura 34, deve ser criada uma tabela que tenha como atributos todos aqueles listados no conjunto de dados e a chave primária será composta por Id_Projeto, que é o atributo identificador dos projetos listados, mais Id_Funcionário, que é o atributo identificador do conjunto de dados de funcionários que estão alocados em cada projeto. A Figura 22 apresenta o resultado dessa transformação.

Id_Projeto	Descrição	Orçamento	Id_Funcionário	Nome_Funcionário
-------------------	------------------	------------------	-----------------------	-------------------------

1	Projeto de BD	20 mil	100	João
1	Projeto de BD	20 mil	200	Pedro
1	Projeto de BD	20 mil	400	Ana
2	Implantação de ERP	100 mil	100	João
2	Implantação de ERP	100 mil	300	Maria
2	Implantação de ERP	100 mi	500	Luiz

Figura 34 – Tabela Projeto na primeira forma normal (1FN).

Fonte: elaborado pelo autor.

Note que os dados apresentados na tabela, normalizada, da Figura 34, são os mesmos da Figura 33, entretanto, todos eles agora fazem parte de uma tabela com apenas um nível de aninhamento, ou seja, não existem grupos repetitivos ou tabelas aninhadas. E, a chave primária é composta pelos atributos `Id_Projeto` e `Id_Funcionário`. Note que `Id_Projeto`, somente, não poderia ser o atributo chave, porque os dados de projeto estão repetidos na tabela, já que vários funcionários trabalham num mesmo projeto. O mesmo ocorre com `Id_Funcionário`, que se repete dentro da tabela, já que os funcionários podem trabalhar em vários projetos.

Note também que existe redundância de dados dentro da tabela, em relação às informações do projeto. O modelo relacional exige que os dados de um projeto se repitam ao longo das linhas para indicar a relação daquele projeto com todos os funcionários que estão associados a ele. Entretanto, não haveria necessidade de que todos os dados do projeto fossem repetidos, apenas a sua identificação (`Id_Projeto`) seria suficiente. Uma das funções da aplicação das regras de normalização é justamente eliminar essa redundância e é o que será visto nas próximas seções. Inicialmente, é preciso introduzir o conceito de dependência funcional.

5.2 Dependência funcional

Considere um esquema genérico de banco de dados com os seguintes atributos (A, B, C, D, ..., Z) e que o banco de dados tenha uma relação $R = (A, B, C, D, \dots, Z)$. O conceito de dependência funcional descreve o relacionamento entre atributos de uma relação da seguinte forma. Sendo A e B atributos da relação R, B é dito funcionalmente dependente de A, se cada valor de A está associado a exatamente a um valor de B.

A Figura 35 (a) apresenta a tabela Funcionário e seus dados. Como é possível perceber, o atributo Departamento lista o departamento da empresa onde o funcionário trabalha. A Figura 35 (b) denota que o atributo Departamento depende funcionalmente de Id_Funcionário, (Id_Funcionário determina Departamento). De acordo com o conceito de dependência funcional apresentado acima, constata-se nos dados da tabela Funcionário que realmente há essa dependência quando é possível identificar que a cada funcionário está associado apenas um departamento. Note que o contrário não é verdade. A cada departamento não há necessariamente um funcionário apenas associado, apesar de, no caso do departamento "RH", haver apenas um funcionário associado, mas, no caso de "Contábil" e "Vendas", há mais de um funcionário associado com cada um, o que evidencia que Id_Funcionário não é funcionalmente dependente de Departamento.

(a)

Id_Funcionário	Nome_Funcionário	Departamento
100	João	Contábil
200	Pedro	Vendas
300	Maria	RH
400	Ana	Contábil
500	Luiz	Vendas



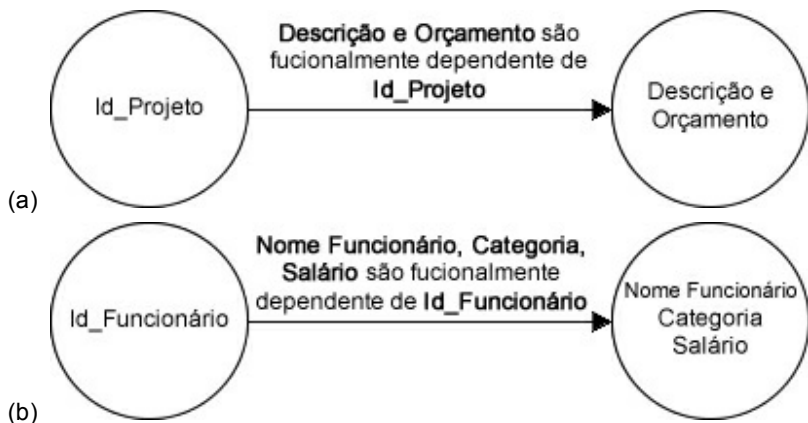
Figura 35 – Exemplo de dependência funcional.

Fonte: elaborado pelo autor.

5.3 Segunda Forma Normal (2FN)

Definição: uma tabela está na segunda forma normal (2FN) se, e somente se, ela estiver na 1FN e os atributos não chaves forem totalmente dependentes da chave primária e não somente de parte dela.

Com base nas definições de 2FN e de Dependência Funcional, expostas logo acima, podemos evidenciar que a tabela Projeto, apresentada na Figura 36, não está na 2FN. Ao analisar as dependências funcionais apresentadas no conjunto de dados da tabela e explicitadas na Figura 36, pode-se perceber que alguns atributos não chave dependem apenas de parte da chave primária da tabela, definida como Id_Projeto mais Id_Funcionário. Apartir dessas evidências, a técnica de normalização determina que a tabela seja subdivida em outras tabelas que respeitem as dependências funcionais identificadas. Portanto, na próxima etapa serão criadas 3 tabelas, uma para Projeto com os atributos que dependem de Id_Projeto; uma para o Funcionário, com os atributos que dependem de Id_Funcionário; e uma ProjetoFuncionário para os atributos que dependem de Id_Projeto e Id_Funcionário. A Figura 37 apresenta as tabelas derivadas nesta fase do processo de normalização.



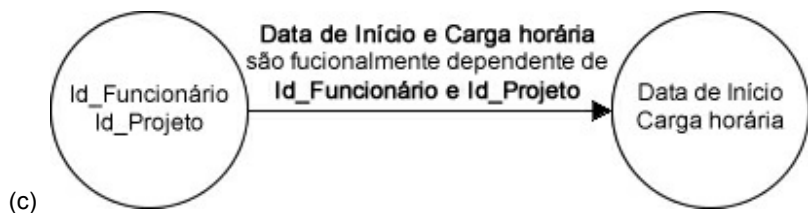


Figura 36 – Dependências funcionais entre os dados da tabela Projeto.

Fonte: elaborado pelo autor.

(a) Tabela Projeto	Id_Projeto	Descrição	Orçamento
	1	Projeto de BD	20 mil
	2	Implantação de ERP	100 mil

(b) Tabela Funcionário	Id_Funcionário	Nome_Funcionário	Categoria	Salário
	100	João	1	3.000
	200	Pedro	2	3.500
	300	Maria	1	3.000
	400	Ana	2	3.500
	500	Luiz	2	3.500

(c) Tabela ProjetoFuncionário	Id_Projeto	Id_Funcionário	Data de Início	Carga Horária
	1	100	10/10/2008	12
	1	200	12/05/2008	8
	1	400	15/07/2008	8
	2	100	13/11/2009	8
	2	300	10/12/2008	12

	2	500	05/10/2009	8
--	---	-----	------------	---

Figura 39 – Relatório com dados para iniciar processo de normalização.

Fonte: elaborado pelo autor.

5.4 Terceira Forma Normal (3FN)

Definição: uma tabela está na terceira forma normal (3FN) se, e somente se, ela estiver na 2FN e todo atributo não chave depende funcionalmente diretamente da chave primária, ou seja, não há dependências entre os atributos não chave.

Na tabela Funcionário, da Figura 35 (b), o atributo Salário, depende, na verdade, de Categoria, que por sua vez depende de Id_Funcionário. Isso ocorre porque, nesse caso, existe uma dependência funcional indireta. É preciso identificar se existe algum caso como esse e seguir o processo de normalização novamente subdividindo as tabelas de acordo com as dependências. A Figura 38 mostra o conjunto de tabelas final de acordo com a passagem à terceira forma normal (3FN).

(d) Tabela Projeto	Id_Projeto	Descrição	Orçamento
	1	Projeto de BD	20 mil
	2	Implantação de ERP	100 mil

(e) Tabela Funcionário	Id_Funcionário	Nome_Funcionário	Categoria
	100	João	1
	200	Pedro	2
	300	Maria	1
	400	Ana	2
	500	Luiz	2

(f) Tabela Categoria	Categoria	Salário
	1	3.000
	2	3.500

(g) Tabela ProjetoFuncionário	Id_Projeto	Id_Funcionário	Data de Início	Carga Horária
	1	100	10/10/2008	12
	1	200	12/05/2008	8
	1	400	15/07/2008	8
	2	100	13/11/2009	8
	2	300	10/12/2008	12
	2	500	05/10/2009	8

Figura 38 – Resultado da passagem à terceira forma normal (3FN).

Fonte: elaborado pelo autor.

Portanto, após uma análise criteriosa, e seguindo as regras estabelecidas na técnica de normalização, o conjunto final de tabelas que compõem o projeto lógico do banco de dados apresenta o menor grau de redundância possível. O que é adequado para evitar problemas em relação à manutenção do banco de dados.

5.5 Um exemplo completo

Acompanhe, por meio do exemplo a seguir, o processo de normalização, passo a passo, a partir de um conjunto de dados não normalizado.

O relatório da Figura 36 apresenta uma lista de pacientes de uma clínica, com alguns dados cadastrais, informações sobre seu plano de saúde e diagnósticos já identificados em suas consultas na clínica. Como é possível observar, o conjunto de dados apresentado não é uma tabela na Primeira Forma Normal (1FN), pois há atributos que não contêm domínio atômico, ou seja, apenas um valor em cada linha. É o caso do atributo Diagnóstico, que contém uma lista de valores em cada linha. A linha 1, por exemplo, tem os valores “Anorexia” e “Bursite”; a linha 2, “Hipoglicemia” e “Pressão Alta”; e assim sucessivamente. Os demais atributos deste conjunto de dados têm domínios atômicos.

Mostre as tabelas geradas nas fases intermediárias (1FN, 2FN), permitindo avaliação parcial caso não consiga completar o exercício. O

resultado deve ser apresentado com o detalhamento das tabelas e seus atributos, indicando chaves primária e estrangeira (quando houver), conforme exemplo a seguir.

Nome	Telefone	Número Matricula	Plano Saúde	Sede Plano Saúde	Diagnostico
Luís	(11)9874567	1	Unimed	Porto Alegre	Anorexia, Bursite
Fernando	(11)89765432	2	Unimed	Porto Alegre	Hipoglicemia, Pressão Alta
Francisco	(34)6678990	3	Amil	São Paulo	Diverticulite, Infecção
Jose	(45)45364567	4	Bradesco	São Paulo	Hipoglicemia, Pressão Alta
Itamar	(68)23234332	5	Sul América	Curitiba	Pressão Alta, Hipoglicemia
Tancredo	(61)56789876	6	Sul América	Curitiba	Infeção, Bursite

Figura 39 – Relatório com dados para iniciar processo de normalização.

Fonte: elaborado pelo autor.

Análise para passagem à 1FN

Conforme mencionado, o conjunto de dados não está normalizado, pois o atributo Diagnóstico não tem domínio atômico, portanto, os dados devem ser reorganizados, distribuindo-se os diferentes valores do atributo de domínio não atômico em quantas linhas forem necessárias. Assim, cada atributo passa a ter domínio atômico. Isso é feito apenas separando os valores do atributo de domínio não atômico em diferentes linhas, sempre relacionando com os dados com quem estava relacionado no conjunto de dados inicial.

Na Figura 40, os valores de Diagnóstico que estavam associados ao paciente “Luis” foram separados em duas linhas, cada uma preenchida com os valores cadastrais de “Luis”; os valores de Diagnóstico de “Fernando” também foram separados em duas linhas com o restante dos

atributos preenchidos com os dados cadastrais de “Fernando”; e assim sucessivamente. Agora, a tabela apresentada na Figura 40 está na 1FN.

Nome	Telefone	Número Matricula	Plano Saúde	Sede Plano Saúde	Diagnostico
Luís	(11)9874567	1	Unimed	Porto Alegre	Anorexia
Luís	(11)9874567	1	Unimed	Porto Alegre	Bursite
Fernando	(11)89765432	2	Unimed	Porto Alegre	Hipoglicemia
Fernando	(11)89765432	2	Unimed	Porto Alegre	Pressão Alta
Francisco	(34)6678990	3	Amil	São Paulo	Diverticulite
Francisco	(34)6678990	3	Amil	São Paulo	Infecção
Jose	(45)45364567	4	Bradesco	São Paulo	Hipoglicemia
Jose	(45)45364567	4	Bradesco	São Paulo	Pressão Alta
Itamar	(68)23234332	5	Sul América	Curitiba	Pressão Alta
Itamar	(68)23234332	5	Sul América	Curitiba	Hipoglicemia
Tancredo	(61)56789876	6	Sul América	Curitiba	Infeção
Tancredo	(61)56789876	6	Sul América	Curitiba	Bursite

Figura 40 – Tabela Usuários na 1FN.

Fonte: elaborado pelo autor.

Para essa tabela, é preciso identificar a sua chave primária, pois é a partir da análise de dependências funcionais entre atributos da tabela 1FN

(chaves e não chaves) que se baseiam todas as formas normais seguintes (2FN e 3FN).

Analisando os valores presentes na tabela 1FN, constata-se que os dois atributos necessários para que uma linha única seja identificada por seus valores são Número Matrícula e Diagnóstico. Portanto, essa será a chave primária composta da tabela. Note, na Figura 41, como os valores das linhas são únicos com nesses dois atributos. Linha 1, “1”, “Anorexia”; linha 2, “1”, “Bursite”; linha 3, “2”, “Hipoglicemia”; linha 4, “2”, “Pressão Alta”; e assim sucessivamente. Como a chave primária da tabela deve ser o conjunto mínimo de atributos que permita que se identifica unicamente uma linha na tabela, esses dois atributos apenas são suficientes.

Nome	Telefone	Número Matrícula	Plano Saúde	Sede Plano Saúde	Diagnostico
Luís	(11)9874567	1	Unimed	Porto Alegre	Anorexia
Luís	(11)9874567	1	Unimed	Porto Alegre	Bursite
Fernando	(11)89765432	2	Unimed	Porto Alegre	Hipoglicemia
Fernando	(11)89765432	2	Unimed	Porto Alegre	Pressão Alta
Francisco	(34)6678990	3	Amil	São Paulo	Diverticulite
Francisco	(34)6678990	3	Amil	São Paulo	Infeção
Jose	(45)45364567	4	Bradesco	São Paulo	Hipoglicemia
Jose	(45)45364567	4	Bradesco	São Paulo	Pressão Alta
Itamar	(68)23234332	5	Sul América	Curitiba	Pressão Alta
Itamar	(68)23234332	5	Sul América	Curitiba	Hipoglicemia
Tancredo	(61)56789876	6	Sul	Curitiba	Infeção

			América		
Tancredo	(61)56789876	6	Sul América	Curitiba	Bursite

Figura 41 – Tabela Usuários com a sua chave primária.

Fonte: elaborado pelo autor.

A descrição da estrutura da tabela apresentada na Figura 41 está na Figura 42; o nome dado à tabela é Usuários, já que é uma tabela que contém dados de pacientes usuários de planos de saúde.

Número Matrícula	Chave primária composta
Diagnóstico	
Nome	
Telefone	
Plano Saúde	
Plano Saúde	

Figura 42 – Estrutura da tabela Usuários.

Fonte: elaborado pelo autor.

Portanto, o projeto do banco de dados em construção por meio da técnica de normalização tem, por enquanto, apenas a tabela Usuários, que está na 1FN. Mas o objetivo da técnica é gerar um banco de dados na Terceira Forma Normal (3FN) e, por isso, é preciso analisar se o projeto está também na 2FN e na 3FN.

Análise para passagem à 2FN

Definição da 2FN, conforme visto neste capítulo: "uma tabela está na segunda forma normal (2FN) se, e somente se, ela estiver na 1FN e os atributos não chaves forem totalmente dependentes da chave primária e não somente de parte dela".

Análise de dependências funcionais para passar para a 2FN: devem ser analisadas as dependências existentes entre os atributos não chave e os atributos que fazem parte da chave primária composta da tabela, para verificar se os atributos não chave dependem de TODA a chave e não

somente de parte dela.

Como o projeto em construção tem apenas a tabela Usuários, a análise é feita sobre os atributos dessa tabela. Os atributos não chave da tabela Usuários são: Nome, Telefone, Plano Saúde e Sede Plano Saúde; e os atributos chave (chave composta) da tabela são: **Número Matrícula** e **Diagnóstico**.

Analizando as dependências entre cada um dos não chave em relação à chave, percebe-se que Nome, Telefone, Plano Saúde e Sede Plano Saúde DEPENDEM APENAS de **Número Matrícula** e não da chave primária toda, que é **Número Matrícula** e **Diagnóstico**. Pode não parecer, mas Sede Plano Saúde depende (indiretamente) de **Número Matrícula**.

Uma dica para verificar se um atributo depende de outro, caso pelo significado do atributo não fique clara a dependência, é verificar se, a cada vez que o atributo determinante se repete na tabela, o atributo dependente se repete também. Isso serve tanto para identificar uma dependência funcional direta como indireta. Veja, na Figura 43, o exemplo de Sede Plano Saúde. Em todas as linhas onde o valor do **Número Matrícula** se repete, o valor de Sede Plano Saúde também se repete. O contrário não é verdade porque a dependência funcional não é comutativa.

Número Matrícula	Sede Plano Saúde
1	Porto Alegre
1	Porto Alegre
2	Porto Alegre
2	Porto Alegre
3	São Paulo
3	São Paulo
4	São Paulo
4	São Paulo
5	Curitiba
5	Curitiba
6	Curitiba
6	Curitiba

Figura 43 – Dados demonstrando a dependência funcional entre Sede Plano Saúde e Número Matrícula.

Fonte: elaborado pelo autor.

A dependência entre Sede Plano Saúde e **Número Matrícula** é indireta porque, de fato, Sede Plano Saúde depende de Plano Saúde, que depende de **Número Matrícula**. Os dados da Figura 44 permitem evidenciar isso.

Número Matrícula	Plano Saúde	Sede Plano Saúde
1	Unimed	Porto Alegre
1	Unimed	Porto Alegre
2	Unimed	Porto Alegre
2	Unimed	Porto Alegre
3	Amil	São Paulo
3	Amil	São Paulo
4	Bradesco	São Paulo
4	Bradesco	São Paulo
5	Sul América	Curitiba
5	Sul América	Curitiba
6	Sul América	Curitiba
6	Sul América	Curitiba

Figura 44 – Dados demonstrando a dependência funcional indireta entre Sede Plano Saúde e Número Matrícula, através de Plano Saúde.

Fonte: elaborado pelo autor.

Com isso, concluímos que a tabela Usuários não está na 2FN, pois há atributos não chave que depende apenas de parte da chave primária. Sendo assim, é preciso dividir a tabela Usuários em tabelas novas, de acordo com as dependências funcionais identificadas.

Nesse caso, haverá uma tabela com os atributos que dependem de **Número Matrícula** e outra tabela com os demais atributos, que, no caso desse exemplo, são aqueles atributos que fazem parte da chave primária da tabela Usuários. Portanto, as tabelas são apresentadas na Figura 45 e 46.

Nome	Telefone	Número Matrícula	Plano Saúde	Sede Plano Saúde
------	----------	------------------	-------------	------------------

Luís	(11)9874567	1	Unimed	Porto Alegre
Fernando	(11)89765432	2	Unimed	Porto Alegre
Francisco	(34)6678990	3	Amil	São Paulo
Jose	(45)45364567	4	Bradesco	São Paulo
Itamar	(68)23234332	5	Sul América	Curitiba
Tancredo	(61)56789876	6	Sul América	Curitiba

Figura 45 – Tabela Usuários na 2FN.

Fonte: elaborado pelo autor.

Número Matrícula	Diagnostico
1	Anorexia
1	Bursite
2	Hipoglicemia
2	Pressão Alta
3	Diverticulite
3	Infeção
4	Hipoglicemia
4	Pressão Alta
5	Pressão Alta
5	Hipoglicemia
6	Infeção
6	Bursite

Figura 46 – Tabela Diagnósticos na 2FN.

Fonte: elaborado pelo autor.

E a definição da estrutura das duas tabelas na 2FN é apresentada nas Figuras 47 e 48.

Número Matrícula	Chave primária
Nome	
Telefone	

Plano Saúde	
Sede Plano Saúde	

Figura 47 – Estrutura da tabela Usuários.

Fonte: elaborado pelo autor.

Número Matrícula	Chave estrangeira para a tabela Usuários	Chave primária composta
Diagnóstico		

Figura 48 – Estrutura da tabela Diagnósticos.

Fonte: elaborado pelo autor.

Análise para passagem à 3FN

Definição da 3FN, conforme visto neste capítulo: uma tabela está na terceira forma normal (3FN) se, e somente se, ela estiver na 2FN e todo atributo não chave depende funcionalmente diretamente da chave primária, ou seja, não há dependências entre os atributos não chave (dependências funcionais indiretas em relação à chave primária).

Análise de dependências funcionais para passar para a 3FN: devem ser analisadas as dependências existentes entre os atributos não chave, para verificar se há dependência entre eles.

Conforme visto antes, Sede Plano Saúde depende de Plano Saúde, logo, essa dependência deve gerar uma nova tabela, retirando-se o atributo Sede Plano Saúde da tabela Usuários. Portanto o novo conjunto de tabelas, todas na 3FN, é apresentado nas Figuras 49, 50 e 51. E a definição da estrutura das tabelas na 3FN é apresentada nas Figuras 52, 53 e 54.

Exemplificando as tabelas com os dados:

Nome	Telefone	Número Matrícula	Plano Saúde
Luís	(11)9874567	1	Unimed
Fernando	(11)89765432	2	Unimed
Francisco	(34)6678990	3	Amil
Jose	(45)45364567	4	Bradesco

Itamar	(68)23234332	5	Sul América
Tancredo	(61)56789876	6	Sul América

Figura 49 – Tabela Usuários na 3FN.

Fonte: elaborado pelo autor.

Plano Saúde	Sede Plano Saúde
Unimed	Porto Alegre
Amil	São Paulo
Bradesco	São Paulo
Sul América	Curitiba

Figura 50 – Tabela Plano de Saúde na 3FN.

Fonte: elaborado pelo autor.

Número Matrícula	Diagnostico
1	Anorexia
1	Bursite
2	Hipoglicemia
2	Pressão Alta
3	Diverticulite
3	Infecção
4	Hipoglicemia
4	Pressão Alta
5	Pressão Alta
5	Hipoglicemia
6	Infecção
6	Bursite

Figura 51 – Tabela Diagnósticos na 3FN.

Fonte: elaborado pelo autor.

Número Matrícula	Chave primária
Nome	

Telefone	
Plano Saúde	Chave estrangeira para a tabela Plano de Saúde

Figura 52 – Estrutura da tabela Usuários.

Fonte: elaborado pelo autor.

Plano Saúde	Chave primária
Sede Plano Saúde	

Figura 53 – Estrutura da tabela Plano de Saúde.

Fonte: elaborado pelo autor.

Número Matrícula	Chave estrangeira para a tabela Usuários	Chave primária composta
Diagnóstico		

Figura 54 – Estrutura da tabela Diagnóstico.

Fonte: elaborado pelo autor.

Note que não houve alteração na tabela Diagnóstico, ela é a mesma gerada na 2FN e na 3FN. Já a tabela Usuários tem uma versão em cada FN. E a tabela Plano de Saúde surgiu apenas na 3FN. Isso será diferente em cada projeto de banco de dados, pois a criação de tabelas ao longo do processo de normalização está relacionada com as dependências funcionais que vão sendo identificadas ao longo da aplicação do processo.

5.6 Denormalização

O esquema de banco de dados final, produzido e validado pela técnica de normalização, pode gerar alguns problemas de desempenho para o SGBD. O principal fator que contribui para isso é o grande número de tabelas no banco de dados e a necessidade de juntar os dados dessas tabelas no momento de realizar algumas consultas. As operações de junção são as que mais geram custo de processamento durante a execução de consultas sobre um banco de dados.

Uma solução para esse problema é fazer um refinamento de um esquema relacional gerado com o objetivo é produzir esquemas de banco

de dados mais eficientes, sem perda de integridade, adicionando atributos e alguma redundância às tabelas. A aplicação de um algoritmo de *denormalização* é um dos passos no projeto de um banco de dados. Esse tipo de alternativa é frequentemente utilizada para sugerir estruturas lógicas alternativas durante o projeto físico do banco de dados, a fase de implementação. O processo é chamado de *denormalização* porque a transformação do esquema do BD pode diminuir o grau de normalização das tabelas.

Existem 4 tipos básicos de *denormalização*:

a. Duas entidades em relacionamentos N:N (muitos para muitos)

- A implementação é feita pela junção da tabela de relacionamento com uma das tabelas envolvidas no relacionamento.
- A tabela resultante dessa construção é composta das chaves primárias de cada uma das entidades associadas.
- Esse tipo de *denormalização* é útil para evitar operações de junção para consultar elementos não chaves.
- O exemplo abaixo apresenta o caso de *denormalização* para as tabelas Funcionário, Figura 55(a), e FuncionárioProjeto, Figura 55(b). Essas duas tabelas foram reunidas em uma nova tabela, apresentada na Figura 55(c). A nova tabela pode servir para consultas em que se precisa saber em que projeto o funcionário trabalha e, ao mesmo, tempo, ter acesso aos dados específicos do funcionário, que são atributos não chave da tabela Funcionário, como seu nome.

(a) Funcionário	Id_Funcionário	Nome_Funcionário	Cat
	100	João	1
	200	Pedro	2
	300	Maria	1
	400	Ana	2

(b) FuncionárioProjeto	Id_Projeto	Id_Funcionário	Data Início
	1	100	10/10.
	1	200	12/05.
	1	400	15/07.
	2	100	13/11.
	2	300	10/12.
	2	500	05/10.

(c) FuncionárioProjeto (denormalizada)	Id_Projeto	Id_Funcionário	Data Início
	1	100	10/10.
	1	200	12/05.
	1	400	15/07.
	2	100	13/11.
	2	300	10/12.
	2	500	05/10.

Figura 55 – Exemplo de *denormalização* envolvendo relacionamento N:N.

Fonte: elaborado pelo autor.

b. Duas entidades em um relacionamento 1:1

- As entidades são implementadas como uma única tabela.
- O objetivo é evitar operações de junção.

c. Entidades com atributos multivalorados

- normalmente armazenados em tabelas separadas, conforme apresentado no capítulo 4, em algumas

situações é mais eficiente implementá-los como uma extensão da tabela principal, pois a redundância pode ser mínima.

d. Atributos derivados (calculados)

- armazenar atributos derivados como parte da tabela para evitar processamentos repetitivos.

5.7 Consequências da aplicação da denormalização

Como a *denormalização*, aumenta a redundância de dados no banco de dados, e, por isso, um dos problemas causados pelo emprego dessa técnica é o uso com mais intensidade do espaço de armazenamento do banco de dados. Também devido ao maior grau de redundância, o custo das operações de alteração do banco de dados também deve aumentar, pois mais dados deverão ser atualizados a cada vez. Consequentemente, o potencial para a perda de integridade do banco de dados é maior, caso alguma atualização não tenha efeito em todas as porções do banco de dados onde deveria ter. Por fim, a manutenção dos programas que usam os dados deve ser feita com cuidado.

Entretanto, apesar desse conjunto de desvantagens, o tempo de acesso aos dados é reduzido por causa do pequeno número de junções necessárias para realizar as operações de consulta ao banco de dados.

Considerando-se, então, que existem vantagens e desvantagens na aplicação dessa técnica, é importante seguir alguns passos para que o custo/benefício seja satisfatório. O conjunto de passos a seguir pode servir como roteiro para a decisão pelo emprego da técnica e definição do projeto final do banco de dados.

- a. Selecionar os processos dominantes que são aplicados sobre o banco de dados em questão, baseando-se nos seguintes critérios:
- frequência de execução;
 - volume de dados acessos;
 - restrições sobre o tempo de resposta.

- b. Definir as novas tabelas, resultantes de junções entre as tabelas definidas no esquema normalizado, e que atendem aos processos identificados no passo anterior.
- c. Analisar o custo total de armazenamento para o novo esquema definido (*denormalizado*) e para o original (normalizado).
- d. Definir o esquema físico que será utilizado com base na comparação feita no passo anterior.

5.8 Recomendações para complementação de estudos referentes aos temas deste capítulo

Para aprofundar os estudos sobre Normalização leia a seguinte referência bibliográfica:

TEOREY, Toby; LIGHTSTONE, Sam; NADEAU, Tom. **Projeto e modelagem de banco de dados**. Tradução de Daniel Vieira. Rio de Janeiro: Elsevier, 2007.

5.9 Resumo do capítulo

Este capítulo apresentou a técnica de Normalização, uma técnica para projeto de banco de dados relacionais, que auxilia na identificação do melhor agrupamento dos dados de forma a atender os requisitos da aplicação. A compreensão dos conteúdos apresentados neste capítulo é importante porque eles estão intimamente ligados ao Modelo Relacional e o domínio de tais conceitos é fundamental para o estudo das linguagens de acesso a banco de dados relacionais que serão estudadas nos próximos capítulos.

¹⁰ O exemplo apresentado neste capítulo é baseado em material didático elaborado originalmente pelo professor Ronaldo dos Santos Mello da UFSC.

CAPÍTULO 6

ÁLGEBRA RELACIONAL

Este capítulo apresenta a Álgebra Relacional, uma linguagem formal para acesso ao banco de dados. O principal motivo de estudar a Álgebra Relacional é a possibilidade de melhor compreender os conceitos do modelo relacional e ter uma base de estudo para linguagens de mais alto nível, como SQL, que será vista no capítulo 6.

A Álgebra Relacional é uma linguagem formal para acesso ao banco de dados e é a base para o desenvolvimento de SQL (*Structured Query Language*), que será abordada no próximo capítulo deste livro.

Segundo (CONNOLLY, 2010), a Álgebra Relacional pode ser descrita como uma linguagem procedural de alto nível. Ela pode ser usada para dizer ao SGBD como construir uma nova relação a partir de relações existentes no banco de dados.

Um dos princípios fundamentais e que precisa ser compreendido em primeiro lugar é que uma expressão descrita na Álgebra Relacional realiza operações sobre relações (tabelas) e o resultado de qualquer operação é uma nova relação (tabela). Essa característica permite que as operações da Álgebra Relacional sejam aninhadas, ou seja, o resultado de uma operação pode ser aplicado como entrada para outra e assim sucessivamente, da mesma forma como nas expressões aritméticas.

As operações da Álgebra Relacional que serão apresentadas neste livro são: Seleção, Projeção, Produto Cartesiano, União, intersecção, Diferença e Junção. Cada uma das operações será apresentada com base em um banco de dados exemplo, que é descrito na Figura 28.

Funcionário			
Chave Primária		Chave Estrangeira que relaciona com Departamento	Chave Estrangeira que relaciona com Categoria
Id_Funcionário	Nome_Funcionário	Departamento	Categoria
100	João	Contábil	1
200	Pedro	Vendas	2

300	Maria	RH	1
400	Ana	Contábil	2
500	Luiz	Vendas	2

Categoria	
Chave Primária	
Categ	Salário
1	3.000
2	3.500

Departamento		
Chave Primária		Chave Estrangeira que relaciona com Funcionário
Departamento	Descrição	Gerente
Contábil	Contabilidade empresarial	100
Vendas	Marketing e vendas	500
RH	Capital Humano e treinamento	300

Projeto		
Chave Primária		
Id_Projeto	Descrição	Orçamento
1	Projeto de BD	20 mil
2	Implantação de ERP	100 mil

ProjetoFuncionário			
Chave Primária			
Chave Estrangeira que relaciona com Projeto	Chave Estrangeira que relaciona com		

	Funcionário		
Id_Projeto	Id_Funcionário	Data de Início	Carga Horária
1	100	10/10/2008	12
1	200	12/05/2008	8
1	400	15/07/2008	8
2	100	13/11/2009	8
2	300	10/12/2008	12
2	500	05/10/2009	8

Figura 56 – Banco de dados (exemplo).

Fonte: elaborado pelo autor.

6.1 Seleção

O operador de seleção recupera as tuplas de uma relação (tabela) que satisfazem determinado critério de busca e produz um subconjunto horizontal da relação.

Notação:

$\sigma_{\langle \text{condição} \rangle} (\langle \text{relação} \rangle)$

σ (sigma) é o símbolo do operador de seleção. A condição pode ser estabelecida como uma expressão relacional, envolvendo os operadores de relacionais: =, < , <= , > , >=. A condição também pode ser uma expressão lógica, envolvendo os operadores lógicos.

A relação pode ser uma tabela original do banco de dados ou uma expressão da Álgebra Relacional.

Exemplos:

Selecionar os funcionários cuja categoria salarial tenha valor igual a

1.

$\sigma_{\text{Categoria} = 1} (\text{Funcionário})$

Id_Funcionário	Nome_Funcionário	Categoria
100	João	1
300	Maria	1

Figura 56 – Resultado para Categoria = 1 sobre a relação Funcionário.

Fonte: elaborado pelo autor.

Selecionar os funcionários do departamento de vendas.

σ Departamento = Vendas (Funcionário)

Id_Funcionário	Nome_Funcionário	Departamento	Categoria
200	Pedro	Vendas	2
500	Luiz	Vendas	2

Figura 57 – Resultado para Departamento = Vendas sobre a relação Funcionário.

Fonte: elaborado pelo autor.

Selecionar os funcionários cuja categoria salarial tenha valor 2 e que são do departamento de vendas.

σ Categoria = 2 e Departamento = Vendas (Funcionário)

Id_Funcionário	Nome_Funcionário	Departamento	Categoria
200	Pedro	Vendas	2
500	Luiz	Vendas	2

Figura 58 – Resultado para Categoria = 2 e Departamento = Vendas sobre a relação Funcionário.

Fonte: elaborado pelo autor.

6.2 Projeção

O operador de projeção recupera determinados atributos de uma relação (tabela), produzindo um subconjunto vertical da relação.

Notação:

$\pi \langle \text{lista_atributos} \rangle (\langle \text{relação} \rangle)$

π (Pi) é o símbolo do operador de projeção. A lista de atributos é a explicitação dos atributos que se deseja no resultado da operação e devem ser separados por vírgulas. A relação pode ser uma tabela original do banco de dados ou uma expressão da Álgebra Relacional.

Exemplos:

Recuperar o nome dos funcionários e o departamento onde trabalham.

$\pi \text{ Nome_Funcionário, Departamento (Funcionário)}$

Nome_Funcionário	Departamento
João	Contábil
Pedro	Vendas
Maria	RH
Ana	Contábil
Luiz	Vendas

Figura 59 – Resultado da projeção de Funcionário em Nome_Funcionário e Departamento.

Fonte: elaborado pelo autor.

6.3 Produto cartesiano

O operador de produto cartesiano multiplica duas relações para definir uma nova que consiste de todas as combinações das tuplas das duas relações.

Notação:

$\langle \text{relação}_1 \rangle \times \langle \text{relação}_2 \rangle$

Tanto relação_1 quanto relação_2 podem ser tabelas originais do banco de dados ou expressões da Álgebra Relacional.

Exemplos:

Recuperar os dados dos funcionários e o valor dos seus salários.

σ Funcionário.Categoria = Categoria.Categ (Funcionário X Categoria)

Para realizar essa consulta é preciso usar a operação de produto cartesiano para colocar os dados das relações Funcionário e Categoria na mesma tabela, já que os dados necessários para resolver o problema estão nessas duas tabelas. A Figura 58 apresenta, inicialmente, o resultado do produto cartesiano.

Funcionário X Categoria

Funcionário. Id_Funcionário	Funcionário. Nome_Funcionário	Funcionário. Departamento	Funcionário. Categoria	Categ Cate
100	João	Contábil	1	1
100	João	Contábil	1	2
200	Pedro	Vendas	2	1
200	Pedro	Vendas	2	2
300	Maria	RH	1	1
300	Maria	RH	1	2
400	Ana	Contábil	2	1
400	Ana	Contábil	2	2
500	Luiz	Vendas	2	1
500	Luiz	Vendas	2	2

Figura 60 – Resultado do produto cartesiano entre Funcionário e Categoria.

Fonte: elaborado pelo autor.

Observe que os dados das duas tabelas, agora, estão agrupados na mesma tabela, mas com todas as combinações possíveis entre eles, inclusive aquelas que não têm resultado significativo algum. Observe também que a tabela resultante do produto cartesiano tem seus atributos nomeados com um prefixo, que é o nome da tabela de onde o atributo vem. Na Figura 60 os atributos Funcionário.Categoria e Categoria.Categ estão em outra cor porque são esses dois atributos que estabelecem a relação entre as duas tabelas originais, pois são justamente a chave

estrangeira de Funcionário e a chave primária de Categoria, respectivamente. E, por fim, veja que as únicas tuplas que tem resultado significativo são aquelas onde o os valores dos atributos que estabelecem a relação entre as duas tabelas são os mesmos, ou seja, as tuplas onde o valor de Funcionário.Categoria é igual a Categoria.Categ. Essas tuplas estão também pintadas de outra cor na Figura 60.

A Figura 61 apresenta a tabela final, resultante da expressão apresentada como resposta para este exemplo.

Funcionário. Id_Funcionário	Funcionário. Nome_Funcionário	Funcionário. Departamento	Funcionário. Categoria	Categ. Categ
100	João	Contábil	1	1
200	Pedro	Vendas	2	2
300	Maria	RH	1	1
400	Ana	Contábil	2	2
500	Luiz	Vendas	2	2

Figura 61 – Resultado de σ Funcionário.Categoria = Categoria.Categ (Funcionário X Categoria).

Fonte: elaborado pelo autor.

Finalmente, caso não se queira que todos os atributos estejam no resultado final, é possível aplicar uma projeção para eliminar atributos.

π Funcionário.Id_Funcionário, (σ Funcionário.Categoria
Funcionário.Nome_Funcionário, Categoria.Categ (Funcionário
Funcionário.Departamento, Categoria))¹¹
Categoria.Categ,
Categoria.Salário

Na expressão apresentada acima, a projeção final, nos atributos Funcionário.Id_Funcionário, Funcionário.Nome_Funcionário, Funcionário.Departamento, Categoria.Categ e Categoria.Salário, produz a tabela apresentada na Figura 62.

Funcionário.	Funcionário.	Funcionário.	Categoria.	Categor
---------------------	---------------------	---------------------	-------------------	----------------

Id_Funcionário	Nome_Funcionário	Departamento	Categ	Salário
100	João	Contábil	1	3.000
200	Pedro	Vendas	2	3.500
300	Maria	RH	1	3.000
400	Ana	Contábil	2	3.500
500	Luiz	Vendas	2	3.500

Figura 62 – Resultado com a eliminação do atributo repetido Funcionário.Categoria.

Fonte: elaborado pelo autor.

Recuperar os dados dos funcionários e os dados dos projetos onde eles trabalham.

π Funcionário.Id_Funcionário, (σ Funcionário.Id_Funcionário =
 Funcionário.Nome_Funcionário, ProjetoFuncionário.Id_Funcionário
 Funcionário.Departamento, e
 Funcionário.Categoria, ProjetoFuncionário.Id_Projeto =
 Projeto.Id_Projeto,
 Projeto.Descrição,
 Projeto.Orçamento,
 ProjetoFuncionário.DataDeInício,
 ProjetoFuncionário.CargaHorária

Novamente nesse exemplo é preciso usar a operação de produto cartesiano para colocar os dados das relações Funcionário, Projeto e FuncionárioProjeto na mesma tabela, já que os dados necessários para resolver o problema estão nessas duas tabelas. A Figura 36 apresenta, inicialmente, o resultado do produto cartesiano. Como pode ser observado, o resultado de um produto cartesiano é, normalmente, grande, pois, como a própria descrição da operação diz, realiza todas as combinações entre os valores das tabelas envolvidas. Na Figura 63 são apresentados, em cor diferente, os atributos (colunas) que fazem a relação entre as tabelas envolvidas no produto e as tuplas (linhas) da tabela que satisfazem a condição de Seleção aplicada sobre esses atributos para obter-se o resultado final. A Figura 64 mostra a tabela final, gerada pela avaliação (execução) completa da expressão de consulta.

Funcionário. Id_Funcionário	Funcionário. Nome_ Funcionário	Funcionário. Departamento	Funcionário. Categoria	Projeto Funcionário. Id_Projeto
100	João	Contábil	1	1
100	João	Contábil	1	1
200	Pedro	Vendas	2	1
200	Pedro	Vendas	2	1
300	Maria	RH	1	1
300	Maria	RH	1	1
400	Ana	Contábil	2	1
400	Ana	Contábil	2	1
500	Luiz	Vendas	2	1
500	Luiz	Vendas	2	1
100	João	Contábil	1	1
100	João	Contábil	1	1
200	Pedro	Vendas	2	1
200	Pedro	Vendas	2	1
300	Maria	RH	1	1
300	Maria	RH	1	1

400	Ana	Contábil	2	1
400	Ana	Contábil	2	1
500	Luiz	Vendas	2	1
500	Luiz	Vendas	2	1
100	João	Contábil	1	1
100	João	Contábil	1	1
200	Pedro	Vendas	2	1
200	Pedro	Vendas	2	1
300	Maria	RH	1	1
300	Maria	RH	1	1
400	Ana	Contábil	2	1
400	Ana	Contábil	2	1
500	Luiz	Vendas	2	1
500	Luiz	Vendas	2	1
100	João	Contábil	1	2
100	João	Contábil	1	2
200	Pedro	Vendas	2	2
200	Pedro	Vendas	2	2

300	Maria	RH	1	2
300	Maria	RH	1	2
400	Ana	Contábil	2	2
400	Ana	Contábil	2	2
500	Luiz	Vendas	2	2
500	Luiz	Vendas	2	2
100	João	Contábil	1	2
100	João	Contábil	1	2
200	Pedro	Vendas	2	2
200	Pedro	Vendas	2	2
300	Maria	RH	1	2
300	Maria	RH	1	2
400	Ana	Contábil	2	2
400	Ana	Contábil	2	2
500	Luiz	Vendas	2	2
500	Luiz	Vendas	2	2
100	João	Contábil	1	2

100	João	Contábil	1	2
200	Pedro	Vendas	2	2
200	Pedro	Vendas	2	2
300	Maria	RH	1	2
300	Maria	RH	1	2
400	Ana	Contábil	2	2
400	Ana	Contábil	2	2
500	Luiz	Vendas	2	2
500	Luiz	Vendas	2	2

Figura 63 – Resultado do produto cartesiano Funcionário X Projeto

Fonte: elaborado pelo autor.

Funcionário. Id_Funcionário	Nome_Funcionário	Departamento	Categoria	Projeto. Id_Projeto
100	João	Contábil	1	1
200	Pedro	Vendas	2	1
400	Ana	Contábil	2	1
100	João	Contábil	1	2
300	Maria	RH	1	2
500	Luiz	Vendas	2	2

Figura 64 – Resultado final gerado pela expressão que recupera os dados dos funcionários e os dados dos projetos onde eles trabalham.

Fonte: elaborado pelo autor.

6.4 União

O operador de União é um operador binário (envolve duas relações como operandos de entrada) e cria uma relação que contém a união de todas as tuplas das duas relações envolvidas. As tuplas duplicadas são automaticamente removidas do resultado. As duas relações operando devem ser compatíveis para a União, ou seja, devem ter o mesmo número de atributos, com aqueles de mesmo domínio dispostos nas mesmas posições.

Notação:

$\langle \text{relação}_1 \rangle \cup \langle \text{relação}_2 \rangle$

Tanto $\langle \text{relação}_1 \rangle$ quanto $\langle \text{relação}_2 \rangle$ podem ser tabelas originais do banco de dados ou expressões da Álgebra Relacional.

Exemplos:

Recuperar a lista de descrições de projetos e de departamentos empregados pela empresa.

$\pi \text{ Descrição (Projeto)} \cup \pi \text{ Descrição (Departamento)}$ ¹³

Para realizar essa consulta pode-se usar o operador de União, entretanto, antes de aplicar a união, é preciso tornar os dois operandos compatíveis. Por isso, as projeções são realizadas em cada tabela, Projeto e Departamento. Veja na Figura 65 o exemplo passo a passo.

• Resultado de $\pi \text{ Descrição (Projeto)}$	Descrição
	Projeto de BD

<ul style="list-style-type: none"> Resultado de π Descrição (Departamento) 	Descrição
	Contabilidade empresarial
	Marketing e vendas
	Capital Humano e treinamento

<ul style="list-style-type: none"> Resultado de π Descrição (Projeto) U π Descrição (Departamento) 	Descrição
	Projeto de BD
	Implantação de ERP
	Contabilidade empresarial
	Marketing e vendas
	Capital Humano e treinamento

Figura 65 – Exemplo de União passo a passo.

Fonte: elaborado pelo autor.

6.5 Diferença

O operador de Diferença é um operador binário (envolve duas relações como operandos de entrada) e cria uma relação com as tuplas que fazem parte da <relação₁>, mas não fazem parte da <relação₂>. As tuplas duplicadas são automaticamente removidas do resultado. As duas relações operando devem ser compatíveis para a Diferença, ou seja, devem ter o mesmo número de atributos, com aqueles de mesmo domínio dispostos nas mesmas posições.

Notação:

$\langle \text{relação}_1 \rangle - \langle \text{relação}_2 \rangle$

Tanto $\langle \text{relação}_1 \rangle$ quanto $\langle \text{relação}_2 \rangle$ podem ser tabelas originais do banco de dados ou expressões da Álgebra Relacional.

Exemplos:

Recuperar a lista de descrições de projetos que não são descrições de departamentos.

π Descrição (Projeto) – π Descrição (Departamento)¹⁴

Para realizar essa consulta pode-se usar o operador de Diferença, entretanto, antes de aplicar a diferença, é preciso tornar os dois operandos compatíveis. Por isso, as projeções são realizadas em cada tabela, Projeto e Departamento. Veja na Figura 39 o exemplo passo a passo.

• Resultado de π Descrição (Projeto)	Descrição
	Projeto de BD
	Implantação de ERP
	Contabilidade Empresarial ¹⁵

• Resultado de π Descrição (Departamento)	Descrição
	Contabilidade empresarial
	Marketing e vendas
	Capital Humano e treinamento

• Resultado de π Descrição (Projeto) – π Descrição (Departamento)	Descrição
	Descrição
	Projeto de BD
	Implantação de

Figura 66 – Exemplo de Diferença passo a passo.

Fonte: elaborado pelo autor.

6.6 Intersecção

O operador de Intersecção é um operador binário (envolve duas relações como operandos de entrada) e cria uma relação com apenas as tuplas que estão presentes nas duas relações envolvidas. As tuplas duplicadas são automaticamente removidas do resultado. As duas relações operando devem ser compatíveis para a Intersecção, ou seja, devem ter o mesmo número de atributos, com aqueles de mesmo domínio dispostos nas mesmas posições.

Notação:

$$\langle \text{relação}_1 \rangle \cap \langle \text{relação}_2 \rangle$$

Tanto $\langle \text{relação}_1 \rangle$ quanto $\langle \text{relação}_2 \rangle$ podem ser tabelas originais do banco de dados ou expressões da Álgebra Relacional.

Exemplos:

Recuperar a lista de descrições de projetos que são também descrições de departamentos.

$$\pi \text{ Descrição (Projeto)} \cap \pi \text{ Descrição (Departamento)}^{16}$$

Para realizar essa consulta pode-se usar o operador de Intersecção, entretanto, antes de aplicar a intersecção, é preciso tornar os dois operandos compatíveis. Por isso, as projeções são realizadas em cada tabela, Projeto e Departamento. Veja na Figura 67 o exemplo passo a passo.

- Resultado de $\pi \text{ Descrição (Projeto)}$

Descrição

Projeto de BD

<ul style="list-style-type: none"> Resultado de π Descrição (Departamento) 	Descrição
	Descrição
	Contabilidade empresarial
	Marketing e vendas
	Capital Humano e treinamento

<ul style="list-style-type: none"> Resultado de π Descrição (Projeto) \cap π Descrição (Departamento) 	Descrição
	Contabilidade empresarial

Figura 67 – Exemplo de Intersecção passo a passo.

Fonte: elaborado pelo autor.

6.7 Junção Natural

O operador de Junção Natural é um operador binário (envolve duas relações como operandos de entrada) e cria uma relação com as tuplas que têm os mesmos valores nos atributos de ligação entre as duas relações, atributos de mesmo nome. Todos os atributos das duas relações envolvidas na junção são levados para a relação resultante, entretanto, apenas uma cópia de cada atributo de ligação (atributos de mesmo nome) aparece no resultado.

Notação:

$\langle \text{relação}_1 \rangle \mid \langle \text{relação}_2 \rangle$

Tanto $\langle \text{relação}_1 \rangle$ quanto $\langle \text{relação}_2 \rangle$ podem ser tabelas originais do

banco de dados ou expressões da Álgebra Relacional.

Exemplos:

Recuperar os dados dos funcionários e o valor dos seus salários.

Funcionário |X| Categoria

Este mesmo problema foi resolvido com Produto Cartesiano e Seleção em um exemplo apresentado anteriormente (a Figura 62 apresenta o exemplo final). Entretanto, é possível usar a operação de Junção Natural, que tem a mesma função do Produto Cartesiano mais a Seleção, já que recupera os dados que tem relação entre as duas tabelas, ou seja, busca as tuplas que têm os mesmos valores nos atributos de ligação. A Figura 41 apresenta o resultado da Junção Natural. Note que o conjunto de atributos do resultado não contém os atributos repetidos (atributos de ligação) e eles também não são nomeados tendo como prefixo o nome da tabela de origem.

Id_Funcionário	Nome_Funcionário	Departamento	Categoria	Salário
100	João	Contábil	1	3.000
200	Pedro	Vendas	2	3.500
300	Maria	RH	1	3.000
400	Ana	Contábil	2	3.500
500	Luiz	Vendas	2	3.500

Figura 68 – Resultado com a eliminação do atributo repetido.

Fonte: elaborado pelo autor.

6.8 Duplicatas

Toda operação da Álgebra Relacional elimina duplicatas automaticamente, ou seja, a tabela resultante de qualquer expressão da Álgebra Relacional não contém linhas (tuplas) iguais.

Por exemplo, a execução da seguinte expressão, envolvendo a tabela Funcionário (Figura 69), produz o resultado apresentado na Figura 70. Veja que as linhas que contém valores repetidos para Departamento não aparecem no resultado (Figura 70).

π Departamento (Funcionário)

Id_Funcionário	Nome_Funcionário	Departamento	Categoria
100	João	Contábil	1
200	Pedro	Vendas	2
300	Maria	RH	1
400	Ana	Contábil	2
500	Luiz	Vendas	2

Figura 69 – Tabela Funcionário.

Fonte: elaborado pelo autor.

Departamento
Contábil
Vendas
RH

Figura 70 – Resultado com a eliminação das linhas com o valor de Departamento repetido.

Fonte: elaborado pelo autor.

O mesmo raciocínio pode ser aplicado para a execução de qualquer operação ou expressão da Álgebra Relacional. Se duas linhas produzidas no resultado forem iguais, somente uma delas permanece no resultado. Note que isso é diferente de como funciona a linguagem SQL, onde é preciso usar uma cláusula específica da linguagem¹⁸, ou método equivalente, para eliminar linhas repetidas do resultado.

6.9 Recomendações para complementação de estudos referentes aos temas deste capítulo

Para aprofundar os estudos sobre Álgebra Relacional leia a seguinte referência bibliográfica:

ULLMAN, Jeffrey D.; WIDOM, Jennifer. **A first course in database**

6.10 Resumo do capítulo

Neste capítulo foi estudada a Álgebra Relacional, que é uma das linguagens que serviu de base para criação da linguagem SQL. O conhecimento sobre o funcionamento de todas as operações da Álgebra Relacional possibilitará a compreensão das estruturas de SQL.

¹¹ A avaliação (execução) de uma expressão da Álgebra Relacional se dá da mesma forma que uma expressão aritmética, ou seja, a ordem de execução é da esquerda para a direita e parênteses forçam precedência na execução das operações, ou seja, operações em parênteses mais internos são realizadas em primeiro lugar. No caso da expressão apresentada, a primeira parte executada é (Funcionário X Categoria). Sobre o resultado gerado no produto cartesiano, é executada a Seleção: σ Funcionário.Categoria = Categoria.Categ. E, por fim, é realizada a Projeção: π Funcionário.Id_Funcionário, Funcionário.Nome_Funcionário, Funcionário.Departamento, Categoria.Categ, Categoria.Salário.

¹² A avaliação (execução) dessa expressão: a primeira parte executada é (Funcionário X ProjetoFuncionário X Projeto). Sobre o resultado gerado no produto cartesiano, é executada a Seleção: σ Funcionário.Id_Funcionário = ProjetoFuncionário.Id_Funcionário e ProjetoFuncionário.Id_Projeto = Projeto.Id_Projeto. E, por fim, é realizada a Projeção: π Funcionário.Id_Funcionário, Funcionário.Nome_Funcionário, Funcionário.Departamento, Funcionário.Categoria, Projeto.Id_Projeto, Projeto.Descrição, Projeto.Orçamento, ProjetoFuncionário.DataDeInício, ProjetoFuncionário.CargaHorária.

¹³ A avaliação desta expressão ocorre da seguinte forma: primeiro é executada a projeção sobre Projeto: π Descrição (Projeto); em seguida, a Projeção sobre Departamento: π Descrição (Departamento); e, por fim, é realizada a União.

¹⁴ A avaliação desta expressão ocorre da seguinte forma: primeiro é executada a projeção sobre Projeto: π Descrição (Projeto); em seguida, a Projeção sobre Departamento: π Descrição (Departamento); e, por fim, é realizada a Diferença.

¹⁵, ¹⁷ Este valor para o atributo Descrição não está presente nas tabelas originais da Figura 26, mas foi inserido aqui para que haja algum resultado na consulta exemplo.

¹⁶ A avaliação desta expressão ocorre da seguinte forma: primeiro é executada a projeção sobre Projeto: π Descrição (Projeto); em seguida, a Projeção sobre Departamento: π Descrição (Departamento); e, por fim, é realizada a Intersecção.

¹⁸ A cláusula DINCT neste caso.

CAPÍTULO 7

SQL

SQL (*Structured Query Language*) é uma linguagem padrão para acesso ao banco de dados. A linguagem SQL é declarativa, pois o código expressa o que se quer que o sistema faça e não como o sistema deve realizar a operação. SQL é baseada na Álgebra Relacional.

A primeira versão de SQL foi desenvolvida pela IBM e chamava-se SEQUEL (CHAMBERLIN, 1974). A linguagem tornou-se um padrão a partir de 1986. A linguagem SQL é padronizada pela *ISO* (*International Organization for Standardization*) e pela *ANSI* (*American National Standards Institute*). Entretanto, possui muitas variações e extensões produzidas pelos diferentes fabricantes de sistemas gerenciadores de bases de dados. Nas documentações dos sistemas específicos constam, normalmente, informações a respeito da sua compatibilidade ao padrão ou não. É sempre importante consultar tais documentos.

O histórico cronológico em relação às versões do padrão SQL inclui as seguintes definições:

- 1986: SQL-86;
- 1989: SQL-89;
- 1992: SQL-92;
- 1999: SQL:1999 (SQL3);
- 2003: SQL:2003;
- 2006: SQL:2006;
- 2008: SQL:2008;
- 2011: SQL:2011.

SQL é subdivida em partes que dão conta de diferentes etapas do processo de desenvolvimento de um projeto de banco de dados. Neste capítulo, serão apresentadas as instruções que fazem parte da porção DDL (*Data Definition Language*) e DML (*Data Manipulation Language*) de SQL. DDL é a parte que reúne as instruções para definição da estrutura do banco de dados, como a criação (CREATE) e remoção (DROP) de tabelas, restrições, índices e visões. DML é a parte que contempla as instruções manipulação, ou seja, inserção (INSERT), atualização (UPDATE), remoção (DELETE), e consulta (SELECT) aos dados do banco de dados.

7.1 Criação de tabelas e restrições

A instrução responsável pela criação das tabelas do banco de dados é CREATE TABLE, que tem a seguinte sintaxe:

```
CREATE TABLE NomeTabela
    ({NomeColuna domínio [NOT NULL]19
        [UNIQUE]
        [DEFAULT opções]
        [CHECK (regra)] [, ...]}
    [PRIMARY KEY (ListaDeColunas),]
    {[UNIQUE (ListaDeColunas)] [, ...]}
    {[FOREIGN KEY (ListaDeColunasForeignKey)
    REFERENCES NomeTabelaReferenciada [(ListaDeColunas)]]
    [, ...]}
    {[CHECK (regra)] [, ...]})
```

A instrução CREATE TABLE, apresentada acima, contempla, além da criação da tabela com suas colunas e domínios, a definição de uma série de restrições de integridade, como a definição das chaves primárias e estrangeiras, além de outras restrições, que serão explicitadas adiante. É importante observar que nem todos os sistemas adotam esse mesmo padrão de sintaxe. Os manuais dos sistemas deverão ser consultados a fim de identificar a sintaxe apropriada.

O CREATE TABLE cria uma tabela chamada NomeTabela com as colunas especificadas. A Figura 71 apresenta as restrições que podem ser definidas a respeito de cada coluna:

[NOT NULL]	Determina que o valor da coluna não pode receber o valor nulo. Se nada for especificado, o valor nulo é permitido.
[UNIQUE]	Determina que o valor da coluna só pode receber valores únicos, ou seja, não pode se repetir ao longo das linhas da tabela.
[DEFAULT opções]	Determina um valor que será

	sempre inserido nesta coluna da tabela, quando um valor não for enviado pela instrução INSERT (que será vista mais adiante).
CHECK (regra)	<p>Permite que seja definida uma condição de restrição (uma regra) que estabelece o conjunto de valores que aquela coluna pode assumir. A regra definida por um CHECK aplica-se somente à coluna aonde foi definida e, portanto, o código da regra só pode mencionar essa coluna. É possível também definir uma cláusula CHECK para toda a tabela, como será visto adiante.</p> <p>Uma cláusula CHECK, quando definida, faz com que, a cada alteração de dados na tabela, o SGBD verifique se o novo valor gerado obedece a mesma, isso vale tanto para inserções, quanto para alterações.</p>

Figura 71 – Restrições que podem ser definidas sobre as colunas de uma tabela em um CREATE TABLE.

Fonte: elaborado pelo autor.

Os demais elementos do CREATE TABLE, são definições de restrições de integridade que se aplicam a toda tabela. Os elementos, que também podem aparecer precedidos por “CONSTRAINT NomeConstraint”, são apresentados na Figura 72.

[PRIMARY KEY (ListaDeColunas),]	Permite a definição da chave primária da tabela. Se a chave for composta, a ListaDeColunas indica os atributos da chave primária, separados por vírgula.
{[UNIQUE (ListaDeColunas)] [, ...]}	Permite a criação de um ou mais índices ²⁰ únicos para a tabela.

<pre>{[FOREIGN KEY (ListaDeColunasForeignKey) REFERENCES NomeTabelaReferenciada [(ListaDeColunas))] [, ...]}</pre>	<p>Permite a definição das chaves estrangeiras da tabela. ListaDeColunasForeignKey determina o atributo, ou conjunto de atributos, que fazem parte da chave estrangeira. NomeTabelaReferenciada indica a tabela onde esses atributos são chave primária. E, opcionalmente, ListaDeColunas lista o nome dos atributos como são definidos na tabela onde são chave primária.</p>
<pre>{[CHECK (regra)] [, ...]}</pre>	<p>Permite que seja definida uma regra de restrição que se aplica a toda a tabela. A regra pode envolver todos os atributos da tabela.</p>

Figura 72 – Restrições que podem ser definidas para toda a tabela em um CREATE TABLE.

Fonte: elaborado pelo autor.

A Figura 73 descreve todas as instruções CREATE TABLE para as tabelas exemplo da Figura 26. Na definição do domínio dos atributos foram usados os tipos de dados definidos no padrão SQL²¹ (CONNOLLY, 2010).

<table><tr><td>Funcionário</td></tr><tr><td>Id_Funcionário</td></tr><tr><td>Nome_Funcionário</td></tr><tr><td>Departamento</td></tr><tr><td>Categoria</td></tr></table>	Funcionário	Id_Funcionário	Nome_Funcionário	Departamento	Categoria	<pre>CREATE TABLE Funcionário (Id_Funcionário NUMERIC NOT NULL, Nome_Funcionário VARCHAR (30) NOT NULL, Departamento VARCHAR (20), Categoria CHAR CHECK (Categoria IN ('1', '2')), PRIMARY KEY (Id_Funcionário), FOREIGN KEY (Departamento) REFERENCES Departamento(Departamento),</pre>
Funcionário						
Id_Funcionário						
Nome_Funcionário						
Departamento						
Categoria						

	FOREIGN KEY (Categoria) REFERENCES Categoria(Categ))
--	---

Está sendo criada a tabela Funcionário, com o atributo Id_Funcionário como sendo sua chave primária. Os atributos Id_Funcionário e Nome_Funcionário não permitem valores nulos. Atributos que fazem parte da chave primária da tabela, ao serem definidos como tal, já são definidos automaticamente como NOT NULL. Para o atributo Categoria está sendo definida uma restrição de integridade do tipo CHECK, que determina que esse atributo somente possa assumir o valor '1' ou '2'. Estão sendo definidas também as duas chaves estrangeiras da tabela: Departamento, que referencia a tabela Departamento; e Categoria, que referencia a tabela Categoria no atributo Categ.

Categoria

Categ

Salário

```
CREATE TABLE Categoria (
    Categ CHAR NOT NULL,
    Salário REAL NOT NULL,
    PRIMARY KEY (Categ))
```

Está sendo criada a tabela Categoria, com o atributo Categ sendo sua chave primária. Também está sendo criado o atributo Salário, que não pode receber o valor nulo.

Departamento

Departamento

Descrição

Gerente

```
CREATE TABLE Departamento (
    Departamento VARCHAR
(20) NOT NULL,
    Descrição VARCHAR (50),
    Gerente NUMERIC,
    PRIMARY KEY
(Departamento),
    FOREIGN KEY (Gerente)
REFERENCES
Funcionário(Id_Funcionário))
```

É criada a tabela Departamento, que tem como chave primária o atributo Departamento. Estão sendo criados também os atributos Descrição e Gerente, sendo que Gerente é a chave estrangeira desta tabela e estabelece uma relação com a tabela Funcionário.

Projeto

```
CREATE TABLE Projeto (
```


Id_Projeto	Id_Projeto NUMERIC NOT NULL,
Descrição	Descrição VARCHAR (50),
Orçamento	Orçamento REAL CHECK (Orçamento >= 0 AND Orçamento <= 150.000),
	PRIMARY KEY (Id_Projeto))

A tabela Projeto é criada e sua chave primária é o atributo Id_Projeto. Também são criados os atributos Descrição e Orçamento, sendo que Orçamento tem uma restrição de integridade, que valida o valor de Orçamento somente se estiver na faixa entre 0 e 150.000.

ProjetoFuncionário	CREATE TABLE ProjetoFuncionário
Id_Projeto	(
Id_Funcionário	Id_Projeto NUMERIC NOT NULL,
Data de Início	Id_Funcionário NUMERIC NOT NULL,
Carga Horária	DataDeInício DATE NOT NULL,
	CargaHorária NUMERIC CHECK(CargaHorária >= 4 AND CargaHorária <= 16),
	PRIMARY KEY (Id_Projeto, Id_Funcionário),
	FOREIGN KEY (Id_Projeto) REFERENCES Projeto,
	FOREIGN KEY (Id_Funcionário) REFERENCES Funcionário)

Está sendo criada a tabela ProjetoFuncionário, que tem sua chave primária composta pelos atributos Id_Projeto e Id_Funcionário. Esses dois atributos são também, de forma individual, chaves estrangeiras nesta tabela. Id_Projeto estabelece a relação com Projeto e Id_Funcionário com Funcionário. Esta tabela também tem uma restrição de integridade definida para o atributo CargaHorário e que determina que o seu valor deve ficar sempre na faixa entre 4 e 16 horas.

Figura 73 – Criação das tabelas do banco de dados exemplo.

Fonte: elaborado pelo autor.

7.2 Alterações na estrutura do banco de dados

A sintaxe padrão de SQL também prevê uma instrução para alteração da estrutura do banco de dados. A instrução ALTER TABLE tem a seguinte sintaxe básica.

```
ALTER TABLE NomeTabela
[ADD [COLUMN] NomeColuna domínio [NOT NULL] [UNIQUE]
    [DEFAULT opção] [CHECK (regra)]]
[DROP [COLUMN] NomeColuna [RESTRICT | CASCADE]]
[ADD [CONSTRAINT [NomeConstraint]] DefiniçãoConstraint]
[DROP [CONSTRAINT NomeConstraint] [RESTRICT | CASCADE]]
[ALTER [COLUMN] SET DEFAULT opção]
[ALTER [COLUMN] DROP DEFAULT]
```

A sintaxe da instrução ALTER TABLE permite que as seguintes alterações sejam feitas na estrutura de uma tabela já criada:

- Adicionar uma nova coluna à tabela (ADD COLUMN);
- Remover uma coluna existente na tabela (DROP COLUMN);
- Adicionar uma nova restrição à tabela (ADD CONSTRAINT);
- Remover uma restrição (DROP CONSTRAINT);
- Definir um valor *default* para uma coluna;
- Remover um valor *default* definido para uma coluna.

Em DROP COLUMN e DROP CONSTRAINT existem as opções RESTRICT e CASCADE. Se RESTRICT é definido, a coluna ou restrição não poderá ser removida caso seja referenciada por algum outro objeto do sistema, por outra tabela, por exemplo. Se a opção CASCADE for definida e outro objeto faz referência ao elemento que está sendo removido, o objeto que faz referência será removido também.

Abaixo é apresentado um exemplo de alteração da tabela Categoria, onde é criado um valor DEFAULT para o atributo Salário:

```
ALTER TABLE Categoria
    ALTER Salário SET DEFAULT 1.000;
```

Abaixo é apresentado um exemplo de alteração da tabela Departamento, onde é o atributo Descrição é removido:

```
ALTER TABLE Departamento  
DROP COLUMN Descrição;
```

7.3 Remoção de tabelas do banco de dados

A instrução do SQL padrão para a remoção de tabelas do banco de dados é a seguinte:

```
DROP TABLE NomeTabela [RESTRICT | CASCADE]
```

Esta instrução remove tanto a estrutura da tabela quanto os seus dados. A opção RESTRICT impede a remoção, caso haja outros objetos que referenciam a tabela sendo removida. E a opção CASCADE procede com a remoção da tabela, assim como de todos os objetos que dependem dela, e assim sucessivamente. É importante tomar muito cuidado ao usar essa opção.

No exemplo abaixo é removida a tabela FuncionárioProjeto.

```
DROP TABLE FuncionárioProjeto
```

7.4 Índices

Índices são estruturas que servem para agilizar o acesso aos dados de um banco de dados. Os índices são criados em função de atributos do banco de dados que são comumente acessados nas consultas. Por exemplo, na tabela de Funcionários, pode ser muito comum as consultas serem realizadas a partir de NomeFuncionário. Nesse caso, seria interessante a criação de um índice por esse atributo. Índices são criados automaticamente para os atributos que compõem a chave primária de uma tabela.

A sintaxe para a criação de índices, normalmente seguida pelas

SGBDs, apesar e de não ser prevista no padrão SQL, é a seguinte.

```
CREATE [UNIQUE] INDEX NomeÍndice  
ON NomeTabela (NomeColuna)
```

Se a cláusula UNIQUE for indicada, o índice somente permitirá valores únicos nas entradas do índice, ou seja, não poderão haver valores repetidos no atributo, ou conjunto de atributos, que compõem o índice. Portanto, é preciso ter cuidado ao criar um índice UNIQUE em tabelas que já contêm valores, pois essa tabela pode conter valores repetidos no atributo para o qual o índice está sendo criado e, nesse caso, o índice não poderá ser criado.

No exemplo abaixo é criado um índice, não único, no atributo Nome_Funcionário da tabela Funcionário.

```
CREATE INDEX ÍndiceNomeFuncionário  
ON Funcionário(NomeFuncionário)
```

A remoção de um índice é realizada pela instrução a seguir.

```
DROP INDEX NomeÍndice
```

O exemplo abaixo remove o índice em NomeFuncionário.

```
DROP INDEX ÍndiceNomeFuncionário
```

7.5 Visões

Visões são tabelas virtuais criadas para facilitar ou garantir mais segurança no acesso aos dados do banco de dados. Uma visão é definida como uma consulta ao banco de dados, portanto, são tabelas virtuais, e podem ser usadas em qualquer instrução SQL de manipulação (que veremos mais adiante). Por outro lado, há algumas restrições em relação às operações de atualização, pois qualquer operação de atualização

realizada sobre uma visão é, na verdade, realizada sobre as tabelas base que a originaram.

Asintaxe para a criação de uma visão é a seguinte:

```
CREATE VIEW NomeVisão [(NovoNomeColuna [, ...])]
AS subconsulta
```

No exemplo abaixo é criada uma visão sobre as tabelas de Funcionários e ProjetosFuncionários, que retorna somente os nomes dos funcionários da empresa e a descrição dos projetos aos quais eles estão vinculados.

```
CREATE VIEW FuncionáriosProjetos (NomeFuncionário, Descrição)
AS SELECT Nome_Funcionário, Descrição
FROM Funcionário F, ProjetoFuncionário PF, Projeto P
WHERE F.Id_Funcionário = PF.Id_Funcionário AND
PF.Id_Projeto = P.Id_Projeto;
```

Uma das vantagens de se ter uma visão desse tipo é facilitar o desenvolvimento de consultas que acessem diversas tabelas do banco de dados. Uma consulta sobre essa visão seria construída como se fosse uma consulta sobre uma única tabela, mesmo que a o SGBD efetivamente vá realizar a consulta sobre várias tabelas. Abaixo está um exemplo de consulta que faz uso da visão FuncionáriosProjetos.

```
SELECT Nome_Funcionário
FROM FuncionáriosProjetos FP
WHERE FP.Descrição = "Projeto de BD";
```

A consulta associada à visão FuncionáriosProjetos será executada quando a consulta acima for submetida no sistema, portanto, ela irá acessar várias tabelas do banco de dados, mesmo que o código da visão apresente apenas um objeto sendo acessado, a visão.

7.6 Inserção de dados

INSERT é instrução responsável pela inserção de dados em uma tabela. A sua sintaxe é a seguinte:

```
INSERT INTO NomeTabela [(ListaDeColunas)]  
VALUES (ListaDeValores)
```

A ListaDeColunas pode ser informada caso seja necessário alterar a ordem ou omitir alguns dos atributos da tabela que está sofrendo a inserção. Caso contrário, essa lista não precisa ser informada.

A Figura 74 apresenta o conjunto de instruções INSERT que realizam a população do banco de dados usado como exemplo neste livro. As tabelas onde as inserções são feitas são aquelas criadas pelas instruções da Figura 71 e estão sendo populadas com os dados apresentados na Figura 26.

Funcionário	<pre>INSERT INTO Funcionário VALUES (100, 'João', 'Contábil', '1'); INSERT INTO Funcionário VALUES (200, 'Pedro', 'Vendas', '2'); INSERT INTO Funcionário VALUES (300, 'Maria', 'RH', '1'); INSERT INTO Funcionário VALUES (400, 'Ana', 'Contábil', '2'); INSERT INTO Funcionário VALUES (500, 'Luiz', 'Vendas', '2');</pre>
Categoria	<pre>INSERT INTO Categoria VALUES (1, 3000); INSERT INTO Categoria VALUES (2, 3500);</pre>
Departamento	<pre>INSERT INTO Departamento VALUES ('Contábil', 'Contabilidade gerencial', 100); INSERT INTO Departamento VALUES ('Vendas', 'Marketing e Vendas', 500); INSERT INTO Departamento</pre>

	VALUES ('RH', 'Capital Humano e treinamento', 300);
Projeto	INSERT INTO Projeto VALUES (1, 'Projeto de BD', 20000); INSERT INTO Projeto VALUES (2, 'Implantação de ERP', 100000);
ProjetoFuncionário	INSERT INTO ProjetoFuncionário VALUES (1, 100, '10/10/2008', 12); INSERT INTO ProjetoFuncionário VALUES (1, 200, '12/05/2008', 8); INSERT INTO ProjetoFuncionário VALUES (1, 400, '15/07-2008', 8); INSERT INTO ProjetoFuncionário VALUES (2, 100, '13/11/2009', 8); INSERT INTO ProjetoFuncionário VALUES (2, 300, '10/12/2008', 12); INSERT INTO ProjetoFuncionário VALUES (2, 500, '05/10/2009', 8);

Figura 74 – População do banco de dados exemplo.

Fonte: elaborado pelo autor.

7.7 Alteração de dados

A sintaxe da instrução UPDATE, responsável pela atualização dos dados das tabelas, é a seguinte:

```
UPDATE NomeTabela
SET NomeColuna1 = Valor1 [, NomeColuna2 = Valor2 ...]
[WHERE condição]
```

O exemplo abaixo apresenta a alteração da descrição de um projeto da tabela Projeto.

```
UPDATE Projeto
SET Descrição = "Projeto de Banco de Dados"
```

WHERE Id_Projeto = 1

Caso uma condição não seja especificada, todas as tuplas da tabela serão alteradas.

7.8 Remoção de dados

A sintaxe da instrução DELETE, responsável pela remoção dos dados das tabelas é a seguinte:

```
DELETE FROM NomeTabela  
[WHERE condição]
```

No exemplo abaixo são removidas as tuplas de ProjetoFuncionário onde o valor do Id_Funcionário é 100.

```
DELETE FROM ProjetoFuncionário WHERE Id_Funcionário = 100;
```

Operações de remoção mais elaboradas podem ser associadas a condições mais complexas. Caso uma condição não seja específica, todos os dados da tabela são removidos. Entretanto, a instrução DELETE remove apenas os dados, a estrutura da tabela permanece inalterada.

7.9 Consultas aos dados

A instrução SQL responsável por realizar consultas ao banco de dados é o SELECT. Uma instrução SELECT pode ser construída a partir de várias cláusulas, o que permite construir consultas para fazer buscas bastante elaboradas e precisas no banco de dados. Assim como na Álgebra Relacional, uma consulta SELECT tem como resultado uma tabela. A sintaxe básica da instrução SQL²² é apresentada a seguir.


```

SELECT [DISTINCT | ALL] { * | [ExpressãoColuna [AS NovoNome]] [,
....]}
FROM NomeTabela [alias] [, ...]
[WHERE condição]
[GROUP BY ListaColunas] [HAVING condição]
[ORDER BY ListaColunas]

```

Para facilitar o entendimento sobre como deve ser estruturada a instrução SELECT, abaixo são explicitados os elementos que devem ser associados às cláusulas apresentadas na sintaxe da instrução.

SELECT	Atributos que farão parte da tabela que constitui o resultado da consulta. São os atributos listados na operação de Projeção (π) da Álgebra Relacional.
FROM	Tabelas de onde os dados serão recuperados. São as tabelas listadas em uma operação de Produto Cartesiano (\times) da Álgebra Relacional.
WHERE	Condição de busca que vai determinar as linhas das tabelas que serão acessadas para produzir o resultado final. É a condição estabelecida junto à operação de Seleção (σ) da Álgebra Relacional.
GROUP BY	Atributos pelos quais se deseja que o resultado seja agrupado.
HAVING	Condição testada sobre os valores agrupados dos atributos listados no GROUP BY.
ORDER BY	Lista de atributos que determina a ordenação do conjunto de dados da tabela resultante da consulta.

7.10 Recuperação de todos os atributos da tabela

É na própria cláusula SELECT que devem ser definidos os atributos que farão parte da tabela resultante da consulta. Os atributos que devem estar no resultado podem ser listados explicitamente ou, caso todas sejam desejados no resultado, podem ser substituídos por um “*”.

Exemplo:

Recuperar os dados dos funcionários.

Como a solicitação de consulta não explicita quais atributos devem compor a tabela resultante, a expressão SELECT pode ser expressa de duas formas:

```
SELECT  Id_Funcionário,  Nome_Funcionário,  Departamento,
Categoria
FROM Funcionário
OU
SELECT *
FROM Funcionário
```

7.11 Eliminação de duplicatas

A eliminação de valores duplicados é feita por meio da Cláusula DISTINCT.

Exemplo:

Recuperar os Id_Funcionário dos funcionários que trabalham nos projetos da empresa.

Consulta sem o uso do DISTINCT:

<pre>SELECT Id_Funcionário FROM ProjetoFuncionário;</pre> <p>Note que há valores de Id_Funcionário repetidos na tabela resultante.</p>	<table><tr><th>Id_Funcionário</th></tr><tr><td>100</td></tr><tr><td>200</td></tr><tr><td>400</td></tr><tr><td>100</td></tr><tr><td>300</td></tr><tr><td>500</td></tr></table>	Id_Funcionário	100	200	400	100	300	500
Id_Funcionário								
100								
200								
400								
100								
300								
500								

Consulta com o uso do DISTINCT:

<pre>SELECT DISTINCT Id_Funcionário FROM ProjetoFuncionário;</pre> <p>Note que agora não há valores de Id_Funcionário repetidos na tabela resultante.</p>	<table><tr><th>Id_Funcionário</th></tr><tr><td>100</td></tr><tr><td>200</td></tr><tr><td>300</td></tr><tr><td>400</td></tr><tr><td>500</td></tr></table>	Id_Funcionário	100	200	300	400	500
Id_Funcionário							
100							
200							
300							
400							
500							

7.12 Retorno de valores calculados

É possível retornar expressões aritméticas no resultado de uma consulta. Para isso basta compor, no SELECT, expressões com os nomes dos atributos e os operadores aritméticos: +, -, * e /.

Exemplo:

Recuperar os dados dos funcionários que atuam nos projetos e a sua carga horária em minutos.

```
SELECT Id_projeto,
Id_Funcionário, DataDeInício,
CargaHorária * 60 AS 'CH em
minutos'
FROM ProjetoFuncionário;
```

Como a carga horária está armazenada na tabela em horas, basta multiplicar por 60 para que o resultado seja expresso em minutos.

Id_Funcionário	DataDeInício	'CH em minutos'
100	10/10/2008	720
200	12/05/2008	480
400	15/07/2008	480
100	13/11/2009	480
300	10/12/2008	720
500	05/10/2009	480

7.13 Funções de Agregação

Em algumas situações é necessário recuperar os dados das tabelas dos bancos de dados e fazer algum tipo de sumarização desses dados, agregá-los de alguma forma. As funções de agregação previstas no padrão SQL são as seguintes:

- **COUNT(NomeColuna):** retorna o número de linhas em uma determinada coluna. Quando usado na forma **COUNT(*)**, as tuplas com o valor nulo presente na tabela não são desconsideradas no momento da contagem. Por outro lado, se for usado com o nome de algum atributo em específico, as tuplas com valor nulo naquele atributo serão desconsideradas da contagem.

Exemplo:

Recuperar o número de funcionários da empresa que atuam em projetos.

```
SELECT COUNT(*) AS Qtde
FROM ProjetoFuncionário;
```

Com **COUNT(*)** com resultado apresenta um funcionário a mais, já que o funcionário 100 trabalha

Qtde
6

em dois projetos.

```
SELECT COUNT(DISTINCT  
Id_Funcionário) AS Qtde  
FROM ProjetoFuncionário;
```

Já com COUNT (DISTINCT
Id_Funcionário) o resultado
apresentado é o correto, contando
apenas uma vez cada funcionário.

Qtde

5

- SUM(NomeColuna): retorna o somatório dos valores em uma determinada coluna (numérica). Os valores nulos que constarem na tabela para o atributo que está sendo somado serão desconsiderados antes da soma ser realizada.

Exemplo:

Recuperar a carga horária total dos empregados da empresa envolvidos em projetos.

```
SELECT SUM(CargaHorária) AS  
Soma  
FROM ProjetoFuncionário;
```

Soma

56

- AVG(NomeColuna): retorna a média dos valores em uma determinada coluna (numérica). Os valores nulos que constarem na tabela para o atributo em que está sendo feita a média serão desconsiderados antes da realização da operação.

Exemplo:

Recuperar a carga horária média dos empregados da empresa envolvidos em projetos.

SELECT AVG(CargaHorária) AS Média FROM ProjetoFuncionário;	Média
	9,3333

- MIN(NomeColuna): retorna o menor valor em uma determinada coluna. Os valores nulos são desconsiderados;

Exemplo:

Recuperar o menor valor de carga horária de um empregado em um projeto.

SELECT MIN(CargaHorária) AS Mínimo FROM ProjetoFuncionário;	Mínimo
	8

- MAX(NomeColuna): retorna o maior valor em uma determinada coluna. Os valores nulos são desconsiderados;

Exemplo:

Recuperar o maior valor de carga horária de um empregado em um projeto.

SELECT MAX(CargaHorária) AS Máximo FROM ProjetoFuncionário;	Máximo
	12

7.14 Condições no WHERE

7.14.1 Operadores relacionais e lógicos

Assim como as condições da operação de Seleção da Álgebra Relacional, as condições estabelecidas no WHERE podem envolver os operadores relacionais (>, >=, <, <=, =, <>) e lógicos (AND, OR e NOT).

Exemplo:

Selecionar os funcionários cuja categoria salarial tenha valor 2 e que são do departamento de vendas.

SELECT * FROM Funcionário WHERE Categoria = '2' AND Departamento = "Vendas";	Id_Funcionário	Nome_Funcionário	Departamento	Cat
	200	Pedro	Vendas	2
	500	Luiz	Vendas	2

7.14.2 Busca de valores dentro de uma faixa de valores

A cláusula Between permite verificar se um atributo possui seus valores dentro de uma faixa de valores a ser testada.

Exemplo:

Selecionar os funcionários que tenham entre 4 e 8 horas de sua carga horária dedicada a pelo menos um projeto.

SELECT Id_Funcionário, CargaHorária FROM ProjetoFuncionário WHERE CargaHorária between 4 and 8;	Id_Funcionário	CargaHorária
	200	8
	400	8
	100	8

7.14.3 Busca por valores dentro de um conjunto

É possível fazer a busca por valores dentro de um conjunto com a cláusula IN. O IN testa se um dado valor encontra-se dentro de uma lista de valores.

Exemplo:

Selecionar os funcionários que trabalham no departamento de Vendas ou no departamento de RH.

SELECT Id_Funcionário, Nome_Funcionário, Depto FROM Funcionário WHERE Departamento IN ("Vendas", "RH");	Id_Funcionário	Nome_Funcionário	Depto
	200	Pedro	Vendas
	300	Maria	RH
	500	Luiz	Vendas

7.14.4 Busca pelo valor nulo

Como o valor nulo é um valor especial, que denota ausência de informação, existe uma cláusula específica para recuperar tuplas que tenham algum de seus atributos com o valor nulo.

Exemplo:

Selecionar os funcionários que não estão vinculados a departamentos (supondo que essa situação seja possível e que haja algum funcionário nesta condição).

SELECT Id_Funcionário, Nome_Funcionário, Depto FROM Funcionário WHERE Departamento IS NULL;	Id_Funcionário	Nome_Funcionário	Depto
	600	Carmen	Null

7.15 Ordenação da tabela resultante de uma consulta

É possível ordenar os valores de uma tabela resultante de uma consulta com a cláusula ORDER BY.

Exemplo:

Selecionar os dados dos funcionários ordenados pelo nome do funcionário.

SELECT * FROM Funcionário ORDER BY Nome_Funcionário;	Id_Funcionário	Nome_Funcionário	Departamento
	400	Ana	Contábil
	100	João	Contábil
	500	Luiz	Vendas
	300	Maria	RH
	200	Pedro	Vendas

É possível ordenar por mais de um atributo e também alterar a ordem para decrescente, acrescentando DESC ao final da lista de atributos.

7.16 Agrupamento

É possível agrupar os dados na tabela resultante, para fazer

algumas sumarizações, por exemplo. A cláusula que permite fazer o agrupamento é GROUP BY.

Exemplo:

Recuperar a quantidade de funcionários que trabalha em cada departamento.

SELECT Departamento, COUNT(*) AS QTDE FROM Funcionário GROUP BY Departamento;	Departamento	QTDE
	Contábil	2
	RH	1
	Vendas	2

Com a cláusula HAVING, associada a um GROUP BY é possível aplicar alguma condição sobre o resultado do agrupamento e com isso filtrar os grupos que estarão no resultado final.

Exemplo:

Recuperar a quantidade de funcionários que trabalha em cada departamento, mas somente para departamentos com 2 funcionários ou mais.

SELECT Departamento, COUNT(*) AS QTDE FROM Funcionário GROUP BY Departamento HAVING COUNT(*) > 1;	Departamento	QTDE
	Contábil	2
	Vendas	2

7.17 Subconsultas

É possível associar uma subconsulta a uma condição posta no WHERE. Da mesma forma como qualquer condição, os operadores relacionais podem ser usados normalmente. Essa é uma das formas de realizar consultas sobre mais de uma tabela do banco de dados.

Exemplo:

Recuperar o Id_Funcionário dos funcionários que trabalham no projeto "Projeto de BD".

SELECT Id_Funcionário FROM ProjetoFuncionário WHERE Id_Projeto = (SELECT Id_Projeto FROM Projeto WHERE Descrição = "Projeto de BD");	<table><tr><th>Id_Funcionário</th></tr><tr><td>100</td></tr><tr><td>200</td></tr><tr><td>400</td></tr></table>	Id_Funcionário	100	200	400
Id_Funcionário					
100					
200					
400					

Algumas outras cláusulas ainda podem ser empregadas para a busca de dados com subconsultas, são elas: ANY, ALL e EXISTS. Cada uma delas será detalhada a seguir.

7.17.1 ANY

Permite a comparação de um elemento com um conjunto (subconsulta) por meio dos operadores relacionais da seguinte forma:

- = **ANY** (*subconsulta*) → mesmo que IN
- > **ANY** (*subconsulta*) → verdadeiro se o atributo comparado for maior do que algum valor de atributo das tuplas resultantes da subconsulta
- < **ANY** (*subconsulta*)
- < > **ANY** (*subconsulta*)

Exemplo:

Recuperar o Id_Funcionário dos funcionários que trabalham em projetos e que tem sua carga horária maior do que pelo menos um outro funcionário, ou seja, o funcionário (ou funcionários) com menor carga horária não deve aparecer no resultado.

<pre>SELECT Id_Funcionário, CargaHorária FROM ProjetoFuncionário WHERE CargaHorária > ANY (SELECT CargaHorária FROM ProjetoFuncionário);</pre> <p>Note que a menor carga horária alocada nos projetos era de 8 horas e esses funcionários não aparecem no resultado.</p>	Id_Funcionário	CargaHorária
	100	12
	300	12

7.17.2 ALL

Permite a comparação de um elemento com todos os elementos de um conjunto (subconsulta) por meio dos operadores relacionais da seguinte forma:

- = **all** (*subconsulta*) → igual a todos
- > **all** (*subconsulta*) → maior que todos
- < **all** (*subconsulta*) → menor que todos
- < > **all** (*subconsulta*) → diferente de todos

Para um valor sair o resultado, a condição deve ser satisfeita para todos os elementos de um conjunto.

Exemplo:

Recuperar o Id_Funcionário dos funcionários que trabalham em

projetos e que tem sua carga horária maior do que todos os funcionários do projeto “Projeto de BD”.

```
SELECT Id_Funcionário,  
CargaHorária, Id_Projeto  
FROM ProjetoFuncionário  
WHERE CargaHorária >  
ALL
```

```
    (SELECT  
CargaHorária  
FROM  
ProjetoFuncionário  
WHERE Id_Projeto  
IN  
    (SELECT  
Id_Projeto  
FROM Projeto  
WHERE  
Descrição="Projeto de  
BD"));
```

Para a execução desse exemplo, foi alterada a carga horária do funcionário 300 para 16 horas. Assim, esse funcionário satisfaz o que é solicitado na consulta, já que sua carga horária fica de fato maior do que a carga horária de todos os funcionários que têm horas alocadas no projeto “Projeto de BD”.

Note que este exemplo fez uso tanto do “> ANY” como do IN, aninhando as subconsultas em dois níveis.

Id_Funcionário	CargaHorária	Id_Projeto
300	16	2

7.17.3 EXISTS

A cláusula EXISTS (ou NOT EXISTS) é empregada junto a subconsultas e somente nessa situação. A cláusula EXISTS resulta em verdadeiro se, e somente se, existe pelo menos uma tupla no resultado da subconsulta. NOT EXISTS é o oposto, retorna verdadeiro se não retornar nenhuma tupla no resultado da subconsulta.

Com essa cláusula é possível realizar consultas sobre mais de uma tabela do banco de dados.

Exemplo:

Recuperar o nome dos funcionários que trabalham em projetos.

```
SELECT Nome_Funcionário  
FROM Funcionário F  
WHERE EXISTS  
    (SELECT *  
     FROM ProjetoFuncionário PF  
     WHERE PF.Id_Funcionário =  
           F.Id_Funcionário);
```

Note que recuperar apenas o nome dos funcionários pela tabela Funcionário não seria suficiente porque podem haver funcionários nessa tabela que não estão alocados em projetos. Portanto, é necessário que seja feita uma verificação se existe relação do funcionário que está sendo recuperado da tabela Funcionário com os funcionários que estão dentro na tabela ProjetoFuncionário, o que é feito pelo EXISTS e pela condição PF.Id_Funcionário =

Nome_Funcionário
João
Pedro
Maria
Ana
Luiz

7.18 UNION

A cláusula UNION realiza a operação de União (U) da Álgebra Relacional. Da mesma forma que na Álgebra Relacional, é necessário que as duas partes envolvidas na União sejam compatíveis para união.

Exemplo:

Recuperar a lista de descrições de projetos e de departamentos empregados pela empresa.

SELECT Descrição FROM Projeto UNION SELECT Descrição FROM Departamento;	Descrição
	Capital Humano e treinamento
	Contabilidade gerencial
	Implantação de ERP
	Marketing e vendas
	Projeto de BD

7.19 INTERSECT

A cláusula INTERSECT realiza a operação de Intersecção (\cap) da Álgebra Relacional. Da mesma forma que na Álgebra Relacional, é necessário que as duas partes envolvidas na Intersecção sejam compatíveis para intersecção.

Exemplo:

Recuperar a lista de descrições de projetos que são também

descrições de departamentos.

<pre>SELECT Descrição FROM Projeto INTERSECT SELECT Descrição FROM Departamento;</pre> <p>Da mesma forma como apresentado na Figura 38, o valor “Contabilidade empresarial” foi adicionado ao banco dados, na tabela Projeto, para permitir que a consulta exemplo tenha algum resultado.</p>	Descrição
	Contabilidade empresarial

7.20 Consultas envolvendo mais de uma tabela

Consultas que envolvem mais de uma tabela podem ser resolvidas, como visto acima, com o uso de subconsultas e algumas cláusulas especiais. Entretanto, quando os atributos do resultado da consulta devem vir de mais de uma tabela, é necessário que seja empregado o conceito de junção. O conceito de junção foi introduzido no capítulo sobre Álgebra Relacional. Uma junção é implementada em SQL, adicionando-se as tabelas no FROM e estabelecendo-se as condições de junção no WHERE, por meio dos atributos de ligação.

Exemplo:

Recuperar os dados dos funcionários e o valor dos seus salários.

SELECT F*, Salário FROM	Id_Funcionário	Nome_Funcionário	Departamento	Catego
	300	Maria	RH	1
	100	João	Contábil	1

Funcionário	500	Luiz	Vendas	2
F. Categoria	400	Ana	Contábil	2
C	200	Pedro	Vendas	2
WHERE				
F.Categoria				
= C.Categ;				
Assim				
como na				
operação				
de junção				
da Álgebra				
Relacional,				
na				
execução				
dessa				
expressão				
SQL, será				
realizado				
um produto				
cartesiano				
entre				
Funcionário				
e Categoria				
e, logo				
após, será				
feita a				
junção pela				
condição				
F.Categoria				
= C.Categ.				

Note que no exemplo anterior as tabelas listadas no FROM receberam um alias (apelido ou nome abreviado para a tabela). Isso facilita a escrita do restante da expressão, pois os nomes abreviados (menores) das tabelas podem ser usados. Além disso, o uso desse recurso é necessário em algumas consultas em que é preciso fazer buscas comparando os dados de uma tabela com outros dados dela própria.

Exemplo:

Recuperar o Id_Funcionário dos funcionários que têm mais horas alocadas em projetos do que o funcionário 200.

<pre>SELECT PF2.Id_Funcionário, PF2.CargaHorária FROM ProjetoFuncionário PF1, ProjetoFuncionário PF2 WHERE PF1.Id_Funcionário = '200' AND PF2.CargaHorária > PF1.CargaHorária; A Carga Horária do funcionário 200 é de 8 horas, portanto, os funcionários com carga horária maior do que 8 estão no resultado.</pre>	Id_Funcionário	CargaHorária
	100	12
	300	16

7.21 Recomendações para complementação de estudos referentes aos temas deste capítulo

Sugere-se que sejam acessados os manuais dos Sistemas Gerenciadores de Bancos de Dados que implementam a linguagem SQL para verificar as diferenças entre os que eles praticam e o padrão apresentado neste capítulo.

7.21 Resumo do capítulo

Neste capítulo foi estudada a linguagem SQL, que é uma linguagem padrão para acesso a banco de dados. É recomendável que os códigos apresentados neste capítulo sejam implementados de fato em algum SGBD como forma de estudar a linguagem.

¹⁹ Na notação adotada neste livro para a apresentação da sintaxe, quando um elemento aparecer entre de colchetes ("[" e "]") significa que ele é opcional, ou seja, pode ou não estar presente em uma criação de tabela. Portanto, de acordo com o que descreve a notação da instrução CREATE TABLE ao lado, somente a definição de pelo menos uma coluna para a tabela é obrigatória, além do próprio CREATE TABLE NomeTabela.

²⁰ Índices são estruturas de indexação que melhoram o desempenho no acesso aos dados de uma tabela.

²¹ Tipos de dados previstos no padrão SQL e empregados neste livro (CONNOLLY, 2010):

Tipo	Declarações em SQL padrão
Boolean	BOOLEAN
Character	CHAR (para um único caracter), VARCHAR (para tamanho variado)
Numérico	NUMERIC, DECIMAL, INTEGER, SMALLINT, FLOAT, REAL, DOUBLE
Data e hora	DATE, TIME, TIMESTAMP

²² As sintaxes SQL apresentadas neste livro foram adaptadas dos originais apresentados em (CONNOLLY, 2010).

REFERÊNCIAS

CHAMBERLIN, D. D. and BOYCE, R. F. SEQUEL: A structured English query language. In **Proceedings** of the 1974 ACM SIGFIDET (Now Sigmod) Workshop on Data Description, Access and Control (Ann Arbor, Michigan, May 01 - 03, 1974). SIGFIDET '74. ACM, New York, NY, 249-264. Disponível em: <<http://doi.acm.org/10.1145/800296.811515>>. Acesso em: 23 jun. 2010.

CHEN, P. P. The entity-relationship model—toward a unified view of data. **ACM Trans. Database Syst.** 1, 1 (Mar. 1976), 9-36. Disponível em: <<http://doi.acm.org/10.1145/320434.320440>>. Acesso em: 23 jun. 2010.

CODD, E. F. A relational model of data for large shared data banks. **Commun. ACM** 13, 6 (Jun. 1970), 377-387. Disponível em: <<http://doi.acm.org/10.1145/362384.362685>>. Acesso em: 23 jun. 2010.

CONNOLLY, Thomas M.; BEGG, Carolyn E. **Database systems: a practical approach to design, implementation, and management**. Harlow: Addison-Wesley, 2010.

ELMASRI, Ramez; NAVATHE, Shamkant B. **Sistemas de Banco de Dados**. São Paulo: Addison-Wesley, 2005.

HEUSER, Carlos Alberto. **Projeto de Banco de Dados**. 5. Ed. Porto Alegre: Sagra Luzzatto, 2004.

SILBERSCHATZ, Abraham; KORTH, Henry F. **Sistema de banco de dados**. Tradução: Daniel Vieira. 5. ed. Rio de Janeiro: Elsevier, 2006.

TEOREY, Toby; LIGHTSTONE, Sam; NADEAU, Tom. **Projeto e modelagem de banco de dados**. Tradução de Daniel Vieira. Rio de Janeiro: Elsevier, 2007.

ULLMAN, Jeffrey D.; WIDOM, Jennifer. **A first course in database systems**. Upper Saddle River: Prentice Hall, 2002.

SOBRE A AUTORA

DENISE BANDEIRA

Graduada pela Pontifícia Universidade Católica do Rio Grande do Sul (1991) e mestre pela Universidade Federal do Rio Grande do Sul (1995). Atualmente, é Professora da Universidade do Vale do Rio dos Sinos. Tem experiência na área de Ciência da Computação, com ênfase em Banco de Dados. É Coordenadora dos cursos de Bacharelado em Sistemas de Informação, modalidades: presencial e a distância.

UNIVERSIDADE DO VALE DO RIO DOS SINOS – UNISINOS

Reitor: Pe. Marcelo Fernandes de Aquino, SJ

Vice-reitor: Pe. José Ivo Follmann, SJ

Diretor da Editora Unisinos: Pe. Pedro Gilberto Gomes



Editora Unisinos

Avenida Unisinos, 950, 93022-000, São Leopoldo, Rio Grande do Sul, Brasil

editora@unisinos.br

www.edunisinos.com.br

© dos autores, 2015

2015 Direitos de publicação da versão eletrônica (em e-book) deste livro exclusivos da Editora Unisinos.

B214b Bandeira, Denise
Banco de dados : (desenvolvimento de aplicações)
[recurso eletrônico] / Denise Bandeira – 2. ed. – São
Leopoldo : Ed. UNISINOS, 2015.
1 recurso online – (EaD)

ISBN 978-85-7431-704-5

1. Banco de dados. I. Título. II. Série.

CDD 005.74

CDU 004.65

Coleção EAD

Editor: Carlos Alberto Gianotti

Acompanhamento editorial: Jaqueline Fagundes Freitas

Revisão: Eliane Tres Crespo

Editoração: Guilherme Hockmüller

A reprodução, ainda que parcial, por qualquer meio, das páginas que compõem este livro, para uso não individual, mesmo para fins didáticos, sem autorização escrita do editor, é ilícita e constitui uma contrafação danosa à cultura. Foi feito depósito legal.

A coleção EaD, de que faz parte este livro, é uma produção da Universidade do Vale do Rio dos Sinos para apoiar os processos de ensino e aprendizagem dos seus cursos de graduação a distância. Entretanto, o uso dessa obra não fica restrito apenas a essa modalidade de ensino, uma vez que pode servir como orientador no estudo de qualquer acadêmico. Os exemplares foram elaborados a partir da experiência de professores de reconhecido mérito acadêmico da Universidade, e traduzem a excelência dos cursos de graduação ofertados na modalidade presencial e a distância da Instituição.



MEMBRO DA
REDE DE
EDITORAS
UNIVERSITÁRIAS
DA AUSJAL
www.ausjal.org

COLEÇÃO

EAD

EDITORA UNISINOS



UNISINOS