Fundamentos de Banco de Dados

Programa de Pós-Graduação para Engenheiros - USIMINAS

Prof. Dr. George Henrique Godim da Fonseca

PUBLICADO PELO PROGRAMA DE PÓS-GRADUAÇAO EM ENGENHARIA DE PRODUÇAO DA UNIVERSIDADE FEDERAL DE OURO PRETO (PPGEP-UFOP) - ICEA/DEENP

PPGEP - UFOP / ICEA (CAMPUS: JOÃO MONLEVADE) - URL: HTTPS://PPGEP.UFOP.BR

ESTRUTURAÇÃO DO DOCUMENTO EM LATEX: **JÚLIO CÉSAR ALVARENGA**
DISCENTE DO PPGEP-UFOP - ÁREA: PESQUISA OPERACIONAL - LINHA DE PESQUISA: MODELAGEM DE SISTEMAS PRODUTIVOS E LOGÍSTICOS (MSLP)

Primeira impressão, Março 2020

Sumário

Parte III

1. Introdução aos Bancos de Dados

Sistemas de Banco de Dados são um componente essencial da vida na sociedade moderna: diversas tarefas que realizamos diariamente envolvem a interação com um bancos de dados. Por exemplo, fazer um depósito no banco, uma reserva de um voo, uma compra no supermercado ou um lançamento de notas. Essas interações são exemplos de acesso a aplicações tradicionais de banco de dados, nas quais a informação armazenada é essencialmente numérica ou textual [2].

Nos últimos anos a proliferação das redes sociais requereram a criação de bancos de dados enormes para armazenar informações não tradicionais, como imagens e vídeos. Novos tipos de sistemas de bancos de dados foram criados para lidar com essa demanda, os sistemas de armazenamento de *big data* ou sistemas NoSQL.

1.1 Conceitos Básicos

Um **banco de dados** (BD) é uma coleção de dados relacionados. Por **dados**, entende-se fatos conhecidos que podem ser armazenados e têm um significado implícito. Por exemplo, considere os nomes, telefones e endereços das pessoas que você conhece. Atualmente, esses dados estão armazenados nos celulares, que têm o seu próprio software para gerenciar esses dados. Um banco de dados tem as seguintes propriedades implícitas [2]:

- representa algum aspecto do mundo real;
- é uma coleção de dados logicamente coerente;
- é projetado, construído e populado para um propósito específico.

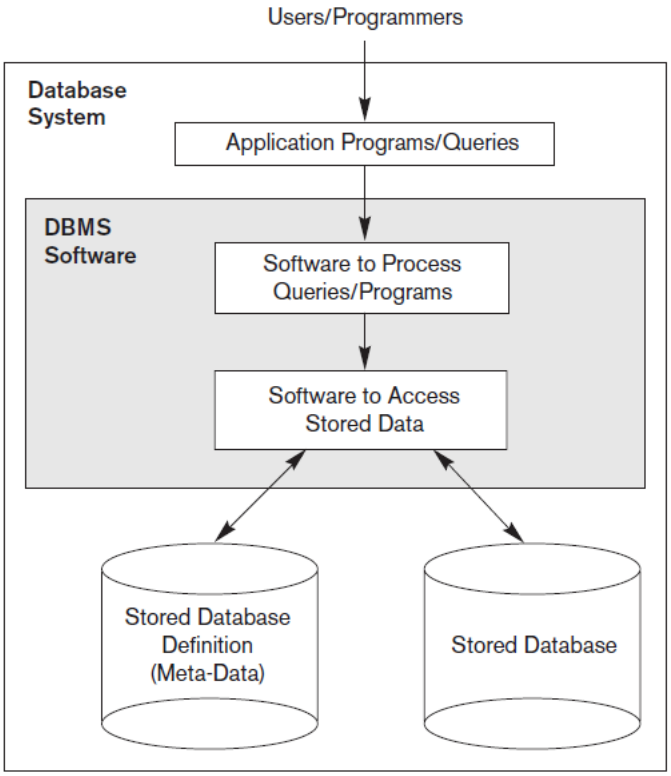
Bancos de dados podem ser de qualquer tamanho ou complexidade. Por exemplo, a lista de contatos mencionada anteriormente pode conter algumas centenas de registros, cada qual com uma estrutura simples. Em contrapartida, um exemplo de banco de dados grande e complexo é o da Amazon.com, que contém dados de 60 milhões de usuários ativos, milhões de livros, jogos, aparelhos eletrônicos e outros itens. Esse banco de dados ocupa mais de 42 terabytes e é armazenado em centenas de computadores [2].

Um **sistema de gerenciamento de banco de dados** (SGBD) é um sistema computadorizado que permite ao usuário criar e manter um banco de dados. Um SGBD é um software de propósito

geral que facilita o processo de definir, construir, manipular e compartilhar bancos de dados entre vários usuários e aplicações. A *definição* de um banco de dados envolve especificar os tipos de dados, estruturas e restrições nos dados a serem armazenados. A definição dos dados e sua informação descritiva é também armazenada no SGBD na forma de um catálogo do banco de dados, chamado **meta-dados**. A *construção* do banco de dados é o processo de armazenar os dados em algum meio controlado pelo SGBD. A *manipulação* do banco de dados inclui funções como consultar, atualizar e gerar relatórios sobre o banco de dados. O *compartilhamento* de um banco de dados permite que múltiplos usuários e programas acessem o banco de dados simultaneamente [2].

Um **programa de aplicação** acessa o banco de dados enviando consultas ou requisições de dados ao SGBD. Uma **consulta** tipicamente seleciona alguns dados para serem recuperados; uma **transação** pode levar a alterações nos dados de um banco de dados. Para completar as definições iniciais, o banco de dados juntamente com seu sistema gerenciador são chamados de **sistema de banco de dados**. A Figura 1.1 apresenta alguns dos conceitos discutidos até agora.

Figura 1.1: Ambiente simplificado de um sistema de banco de dados.



Fonte: [2]

■ **Exemplo 1.1** Consideraremos um contexto simples em que muitos estão familiarizados, um banco de dados UNIVERSIDADE para manter informações sobre alunos, disciplinas, professores e notas em um ambiente universitário. Esse banco de dados será considerado ao longo do curso. A Figura 1.2 apresenta uma parte da estrutura do banco de dados e alguns registros de exemplo.

Essa porção do banco de dados é organizada em cinco tabelas, onde cada qual armazena **registros** do mesmo tipo. Para *definir* esse banco de dados é necessário definir a estrutura dos registros em cada tabela especificando diferentes tipos de **elementos de dados** para serem armazenados em cada registro. É necessário ainda definir um **tipo de dados** para cada coluna da tabela (ou atributo). Por exemplo, o nome na tabela ALUNO é uma cadeia de caracteres, já a nota na tabela OFERTA é

Figura 1.2: Um banco de dados que armazena informações sobre alunos, disciplinas e ofertas de disciplina.

PROFESSOR		
CPF	Nome	Salario
10120214450	Lucas Menezes	5420.00
11040540330	Bruno Santos	4200.00
10560930210	Marcos Lima	7150.00
10010220470	Elton Cardoso	6500.00
10190267390	Rafael Silva	2500.00
10432029020	Fernando Vaz	3880.00

ALUNO			
Matricula	Nome	DataNascimento	Curso
1918023	Camila Braga	12/03/1999	COM
1918142	João Lage	20/11/2001	PRO
1828201	Pedro Santos	02/09/2000	COM

DISCIPLINA		
Codigo	Nome	Horas
CEA160	Cálculo Diferencial e Integral I	60
CSI030	Programação de Computadores I	60
CSI440	Banco de Dados I	60
CSI443	Matemática Discreta	60
CSI488	Algoritmos e Estruturas de Dados I	60

PREREQUISITO	
CodigoDisciplina	CodigoPrerequisito
CSI440	CSI443
CSI440	CSI488
CSI488	CSI030

OFERTA							
Codigo	CodigoDsciplina	Semestre	Ano	Nota	Frequencia	Aluno	Professor
1	CSI030	1	2019	8.6	90%	1918023	10120214450
2	CSI030	1	2019	7.4	95%	1918142	10120214450
3	CSI030	1	2019	2.5	50%	1828201	10120214450
4	CSI440	1	2019	4.7	75%	1918142	11040540330
5	CSI160	1	2019	9.5	100%	1828201	10560930210
6	CSI443	1	2019	7.7	82%	1918023	10010220470
7	CSI030	2	2019	6.0	90%	1828201	10432029020
8	CSI440	2	2019	6.7	85%	1918142	10190267390
9	CSI160	2	2019	8.2	92%	1918023	10560930210
10	CSI443	2	2019	9.0	100%	1828201	10010220470

Fonte: Elaborado pelo autor

um número decimal.

Para *construir* o banco de dados, são armazenados dados para representar cada estudante, disciplina, oferta de disciplina, atribuição de nota e pre-requisito. Observe que várias tabelas podem estar relacionadas. Por exemplo, a tabela pre-requisito relaciona um elemento da tabela disciplina a outro da mesma tabela. A maioria dos bancos de dados apresentam **relacionamentos** entre suas tabelas.

A *manipulação* do banco de dados envolve consultas e atualização. Exemplos de consultas são:

- recuperar as disciplinas ministradas pelo Prof. Bruno Santos;
- selecionar os alunos que tirar nota igual ou superior a 6.0 em 'Matemática Discreta';
- listar os pre-requisitos da disciplina 'Banco de Dados I'.

Exemplos de atualização incluem:

- atualizar o curso do aluno Pedro Santos para PRO;
- criar uma nova disciplina chamada 'Engenharia de Software';
- alterar a nota de João Lage em 'Programação de Computadores I' para 8.0.

■

1.2 Características da Abordagem de Bancos de Dados

Várias características distinguem a abordagem de banco de dados da antiga abordagem de escrever programas personalizados para acessar dados em arquivos. Na abordagem tradicional de **processamento de arquivos**, cada usuário define e implementa os arquivos necessários para o seu software como parte da programação da aplicação. Isso pode levar a problemas como desperdício de armazenamento, redundância de trabalho para manter os dados atualizados e maiores custos no desenvolvimento da aplicação [2].

Na abordagem de banco de dados, um repositório único mantém os dados que são definidos apenas uma vez e acessados por uma variedade de usuários através de consultas e transações. As principais características da abordagem de banco de dados frente à abordagem de processamento de arquivos são dadas nas seções a seguir [2].

1.2.1 Natureza Auto-Descritiva de um Sistema de Banco de Dados

Uma característica fundamental da abordagem de banco de dados é que um sistema de banco de dados não possui apenas o banco de dados mas também uma definição completa da estrutura do banco de dados e suas restrições. Essa definição é armazenada num catálogo de **meta-dados** do banco de dados.

Um SGBD não é programado para uma aplicação específica de banco de dados. Dessa forma, o SGBD se utiliza dos meta-dados para conhecer a estrutura dos arquivos em um BD, como o tipo e o formato dos dados que acessará. Já no processamento tradicional de arquivos, a definição dos dados é tipicamente parte dos próprios das aplicações. Assim, esses programas são construídos para trabalhar com uma única base de dados específica.

1.2.2 Isolamento entre Programa e Dados

No processamento de arquivos tradicional, a estrutura dos arquivos de dados é embutida nas aplicações, assim, qualquer mudança na estrutura de um arquivo pode requerer alterações em todos os programas que acessar tal arquivo. Em contrapartida, aplicações que acessam SGBDs não precisam de tais mudanças na maioria dos casos.

Por exemplo, suponha que se queira adicionar a data de nascimento nos dados do estudante. Ao adicionar esse dado um programa que faz processamento de arquivos teria que ser alterado para compactuar com esse novo formato dos dados. Já programas que acessam SGBDs não teriam que ser alterados. Basta alterar a descrição da tabela ESTUDANTE incluindo a data de nascimento.

1.2.3 Suporte a Múltiplas Visões dos Dados

Um banco de dados tem tipicamente muitos tipos de usuários, onde cada qual pode requerer uma **visão** diferente do BD. Uma visão pode ser um subconjunto dos dados ou conter dados virtuais que são derivados dos arquivos do banco de dados mas não são explicitamente armazenados. Por exemplo, um usuário do banco de dados da Figura 1.2 pode estar interessado apenas em uma transcrição dos dados relacionados a cada estudante.

1.2.4 Compartilhamento de Dados e Processamento de Transações Multiusuário

Um SGBD permite que múltiplos usuários acessem o banco de dados ao mesmo tempo. Assim, os SGBDs têm que implementar o **controle de concorrência** para assegurar que quando vários usuários tentam atualizar os mesmos dados isso seja feito de forma controlada. Por exemplo, quando vários clientes tentam reservar um assento no mesmo voo ao mesmo tempo o SGBD deve assegurar que cada cliente receba um assento distinto.

1.3 Vantagens ao Utilizar um SGBD

Nessa seção são discutidas algumas vantagens em se utilizar um SGBD e as capacidades que um bom SGBD deve possuir. Essas capacidades são em adição às quatro características principais discutidas na Seção 1.2.

1.3.1 Controle de Redundância

A **redundância** ao armazenar os mesmos dados múltiplas vezes leva a vários problemas. Primeiro, é necessário uma única atualização lógica - como adicionar os dados de um novo estudante - múltiplas vezes: uma para cada arquivo em que dados dos estudantes estão gravados. Isso leva a uma *duplicação do esforço*. Segundo, *espaço de armazenamento é desperdiçado* quando o mesmo dado é armazenado mais de uma vez, o que pode ser crítico para bancos de dados grandes. Terceiro, arquivos que representam os mesmos dados podem estar *inconsistentes*.

Na abordagem de BD as visões de diferentes tipos de usuários são integradas no projeto do banco de dados. Idealmente, deve-se ter um projeto de banco de dados que armazena cada item lógico de dados em *apenas um lugar* no BD. Isso é conhecido como **normalização de dados**, e assegura consistência e economiza espaço de armazenamento.

1.3.2 Restrição ao Acesso Não-Autorizado

Quando múltiplos usuários compartilham uma grande base de dados, é comum que muitos usuários não devam ter acesso a toda a informação no banco de dados. Por exemplo, dados financeiros como salários e bônus são comumente considerados confidenciais e apenas pessoas autorizadas podem acessar tais dados. Adicionalmente, alguns usuários podem ter permissão apenas para leitura dos dados enquanto outros podem fazer alterações. Um SGBD deve prover um **subsistema de segurança e autorização**, o qual o **administrador do banco de dados** usa para criar contas e especificar restrições de acesso. Então, o SGBD deve assegurar essas restrições automaticamente.

1.3.3 Prover Armazenamento Persistente para Objetos de Programas

Bancos de dados podem ser usados para prover armazenamento **persistente de objetos** de um programa e estruturas de dados. Linguagens de programação tipicamente têm estruturas de dados complexas. Os valores das variáveis dos programas são descartados assim que o programa termina, a não ser que o programador explicitamente armazene eles em arquivos permanentes, que envolve converter essas estruturas complexas para um formato adequado ao armazenamento em arquivo. Quando é necessário ler esses dados, o programador deve converter os dados do formato de arquivo para a estrutura de dados utilizada. SGBDs são compatíveis com as linguagens de programação mais populares, como Java, Python e C++ e realizam automaticamente todas as conversões necessárias.

1.3.4 Prover Estruturas de Armazenamento e Técnicas de Busca

Sistemas de bancos de dados devem prover capacidades para executar consultas e atualizações eficientemente. Dado que o banco de dados é armazenado em disco, o SGBD deve prover estruturas de dados especializadas e técnicas de busca para acelerar a busca no disco para os registros de interesse. Arquivos auxiliares chamados **índices** são frequentemente utilizados para esse propósito. Índices são tipicamente baseados em uma estrutura de dados em árvore ou hash que são adequadamente modificadas para busca em disco.

1.3.5 Prover Backup e Recuperação

Um SGBD deve prover mecanismos para recuperar de falhas de software ou de hardware. O **subsistema de backup e recuperação** de um SGBD é responsável para tal. Por exemplo, se um computador desliga no meio de uma transação complexa de atualização, o sistema de recuperação

é responsável por assegurar que o banco de dados é restaurado para o estado que estava antes da transação começar a executar.

1.3.6 Representar Relacionamentos Complexos entre Dados

Um banco de dados deve incluir numerosas variedades de dados que são inter-relacionados de muitas formas. Por exemplo, considere o banco de dados da Figura 1.2. O estudante 'Pedro' está relacionado com três registros na tabela OFERTA. Similarmente, cada registro em OFERTA é relacionada com uma disciplina e um professor.

1.3.7 Impor Restrições de Integridade

A maioria das aplicações de banco de dados têm **restrições de integridade** que deve ser atendidas pelos dados. Um SGBD deve prover capacidades para definir e impor essas restrições. A restrição de integridade mais simples envolve especificar um tipo de dados para cada item de dados. Por exemplo, podemos especificar que o tipo de dados do campo período de um estudante é um número inteiro.

Um tipo mais complexo de restrição que ocorre frequentemente envolve especificar que um registro em um arquivo deve ser relacionado a registros em outros arquivos. Por exemplo, na Figura 1.2, deve-se especificar que cada registro de OFERTA seja relacionada a uma DISCIPLINA. Isso é conhecido como restrição de **integridade referencial**. Outro tipo de restrição especifica a unicidade em valores de itens de dados, como cada registro de disciplina deve ter um código único. Isso é conhecido como restrição de **chave** ou **unicidade**.

1.3.8 Implicações Adicionais na Abordagem de Banco de Dados

A seguir são apresentadas algumas implicações adicionais da abordagem de banco de dados que pode beneficiar a maioria das organizações:

Impor Padrões: A abordagem de banco de dados permite ao administrador definir e impor padrões dentre os usuários de banco de dados em uma organização grande. Isso facilita a comunicações e cooperação entre vários departamentos, projetos e usuários na organização.

Tempo de desenvolvimento de Aplicação Reduzido: Uma das principais características da abordagem de banco de dados é que o desenvolvimento de uma nova aplicação - como a recuperação de certos dados do banco de dados para um relatório - leva muito pouco tempo.

Flexibilidade: Pode ser necessário mudar a estrutura do banco de dados ao se mudarem os requisitos. Por exemplo, um novo grupo de usuários pode precisar de informações não presentes no banco de dados. Em resposta, pode ser necessário adicionar uma nova tabela aos banco de dados, ou ampliar os elementos de dados existentes.

Informação Atualizada Disponível: Um SGBD torna o banco de dados disponível para todos os usuários. Assim que a atualização feita por um usuário é aplicada ao banco de dados, todos os demais usuários têm acesso aos dados atualizados.

Economia de Escala: A abordagem de SGBD permite a consolidação de dados e, consequentemente, a redução na quantia de sobreposição entre atividades do pessoal de TI em diferentes projetos bem como a redundância entre aplicações.

1.4 Exercícios

Exercicio 1.1 Defina os seguintes termos: dados, banco de dados, sistema gerenciador de banco de dados, meta-dados, programa de aplicação, sistema de banco de dados. ■

Exercicio 1.2 Apresente as vantagens do uso de um SGBD invés do processamento tradicional de arquivos. ■

Exercicio 1.3 Apresente alguns exemplos de consultas informais que podem ser feitas ao banco de dados da Figura 1.2. ■

Especifique todos os relacionamentos entre os registros do banco de dados apresentado na Figura 1.2.

2. Modelo Entidade-Relacionamento

Nesse capítulo são apresentados os conceitos de modelagem do **modelo entidade-relacionamento (ER)**, que é um modelo conceitual popular de alto-nível. Esse modelo é frequentemente usado para o projeto conceitual de aplicações de banco de dados e muitas ferramentas de projeto de BD empregam esses conceitos. É apresentada ainda uma notação diagramática associada ao modelo ER, conhecida como **diagramas entidade-relacionamento**.

2.1 Uma Aplicação Simples de Banco de Dados

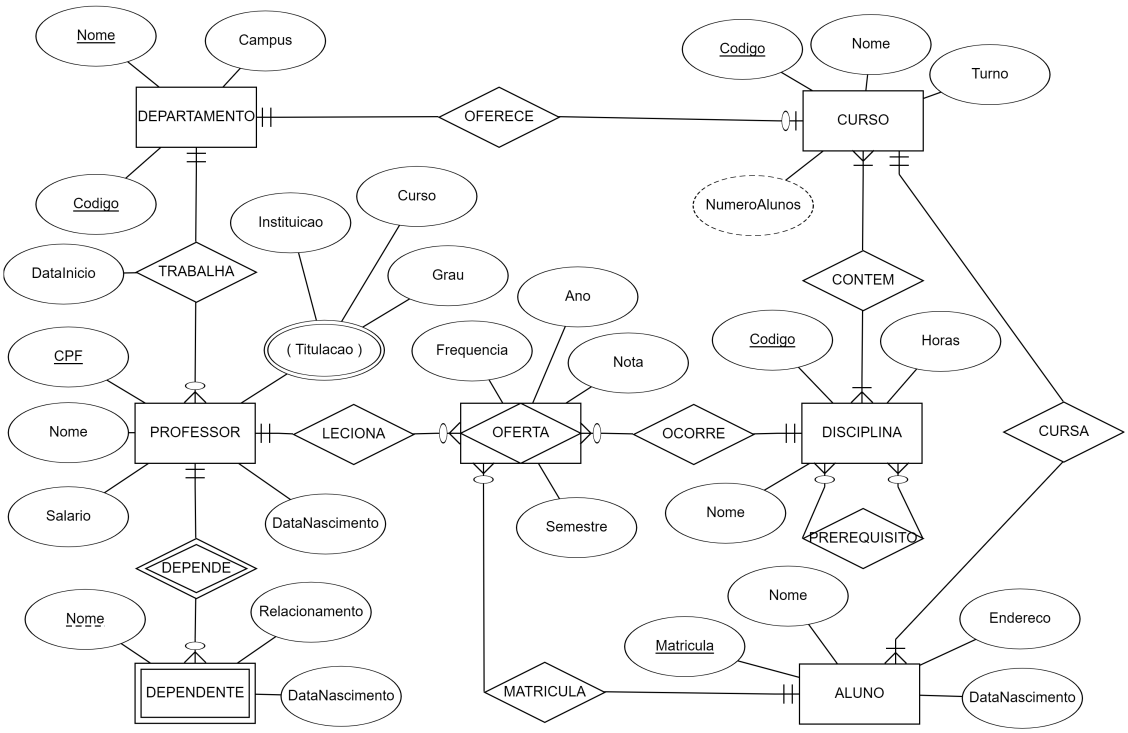
Nessa seção é descrita uma aplicação de banco de dados simples, chamada UNIVERSIDADE, que servirá para ilustrar os conceitos básicos do modelo ER. O banco de dados UNIVERSIDADE mantém dados dos alunos, professores, disciplinas, cursos e departamentos da universidade. Suponha que após a fase de coleta e análise de requisitos os projetistas do banco de dados provêm a seguinte informação sobre o mini-mundo - a parte da universidade que deve ser representada no banco de dados.

- A universidade é organizada em departamentos. Cada departamento tem um nome único, um código único e um endereço. Um departamento emprega vários professores, cuja data de início no departamento deve ser armazenada.
- Um curso é oferecido por um departamento e é composto por um conjunto de disciplinas. Para o curso deve-se armazenar seu código único, seu nome e seu turno. Para a disciplina deve-se armazenar seu código único, seu nome, sua descrição e seus pré-requisitos, que são outra(s) disciplina(s). Uma disciplina pode também fazer parte de vários cursos.
- O banco de dados deve armazenar o nome, CPF (único), data de nascimento, titulação e salário dos professores. Já para os alunos requer-se o armazenamento do nome, matrícula, data de nascimento, curso e endereço. Um professor é associado a um único departamento para o qual trabalha.
- O professor pode ter ainda dependentes, para os quais deve ser armazenado o nome, data de nascimento e relacionamento do dependente com o professor.
- Uma disciplina é ofertada por um professor e nessa oferta diversos alunos podem se matricular.

Para cada disciplina ofertada é necessário armazenar o ano e o semestre em que foi ofertada. Para cada aluno que participa dessa oferta de disciplina é necessário armazenar a nota e a frequência obtidas.

A Figura 2.1 apresenta como o esquema para essa aplicação de banco de dados pode ser representado por meio de um diagrama entidade-relacionamento. O processo passo-a-passo para a geração desse diagrama bem como os conceitos associados serão apresentados nas próximas seções.

Figura 2.1: Diagrama ER para o banco de dados UNIVERSIDADE.



2.2 Tipos e Conjuntos de Entidades, Atributos e Chaves

O modelo ER descreve entidades, atributos e relacionamentos.

2.2.1 Entidades e Atributos

O conceito básico que um modelo ER representa é uma **entidade**, que é uma coisa ou objeto no mundo real com uma existência independente. Uma entidade pode ser um objeto com existência física (por exemplo um carro, pessoa ou casa) ou um objeto com existência conceitual (como uma empresa, curso universitário ou projeto). Cada entidade tem **atributos** - as propriedades que o descrevem [2]. Por exemplo, uma entidade PROFESSOR pode ser descrita por sua idade, CPF, nome, titulação e salário.

Vários tipos de atributo ocorrem em um modelo ER: simples ou composto, valor-único ou multi-valorado e armazenado ou derivado.

Atributos Simples e Compostos

Atributos compostos podem ser divididos em subpartes menores. Por exemplo, o atributo Titulação de um PROFESSOR por ser subdividido em Grau, Curso e Instituição. Atributos que não podem

ser divididos são chamados **atributos simples**, ou atômicos. Atributos compostos podem ainda formar uma hierarquia. O atributo Curso por exemplo, pode ainda ser subdividido em área de concentração e linha de pesquisa.

Atributos de Valor-Único e Multi-Valorados

A maioria dos atributos tem um valor único para uma entidade em particular, esses atributos são chamados de **valor-único**. Um exemplo de atributo de valor-único é o salário de um PROFESSOR. Em alguns casos um atributo pode ter um conjunto de valores para a mesma entidade. Por exemplo, um PROFESSOR pode ter mais de uma titulação. Esse tipo de atributo é chamado **multi-valorado**.

Atributos Armazenados e Derivados

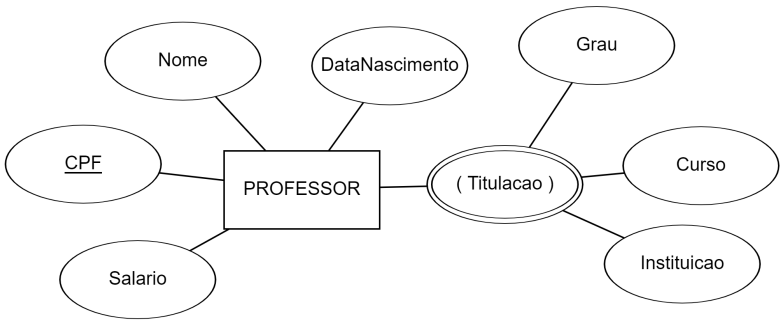
Em alguns casos, dois ou mais atributos são relacionados, como idade e a data de nascimento de uma pessoa. Para a entidade ALUNO o atributo idade pode ser determinado pela data atual em conjunto com sua DataNascimento. O atributo idade é então chamado **atributo derivado** pois é derivável da DataNascimento do ALUNO, que é chamado **atributo armazenado**. Alguns atributos podem ainda ser derivados de entidades relacionadas, como o número de alunos de uma entidade CURSO pode ser derivado contando o número de alunos que cursam o curso.

2.2.2 Tipos e Conjuntos de Entidades, Chaves e Domínios de Atributos

Tipos e Conjuntos de Entidades

Um **tipo de entidade** define uma coleção (ou conjunto) de entidades que têm os mesmos atributos. A coleção de todos os elementos de um tipo de entidade é chamada de **conjunto da entidade**. Um tipo de entidade é representado no modelo ER como um retângulo encapsulando seu nome. Nomes de atributos são encapsulados em forma oval e ligados à entidade correspondente por uma linha reta. Atributos compostos são ligados aos seus atributos originários por linhas retas. Atributos multi-valorados são exibidos com ovais duplos [2]. A Figura 2.2 apresenta um entidade PROFESSOR representada nessa notação.

Figura 2.2: Entidade PROFESSOR representada no modelo ER.



Atributos Chave

Uma importante restrição aos registros de uma entidade é a **chave** ou **restrição de unicidade** em atributos. Uma entidade comumente tem um ou mais atributos que têm que ser distintos para cada registro dessa entidade. Esse atributo é chamado de **atributo chave** e pode ser usado para identificar um registro unicamente [2]. Por exemplo, para a entidade ALUNO o atributo chave é a Matricula. Às vezes mais de um atributo juntos formam a chave, por exemplo a entidade DEPARTAMENTO, que tem como chaves o nome e o número. Um atributo chave é identificado no modelo ER com o nome sublinhado.

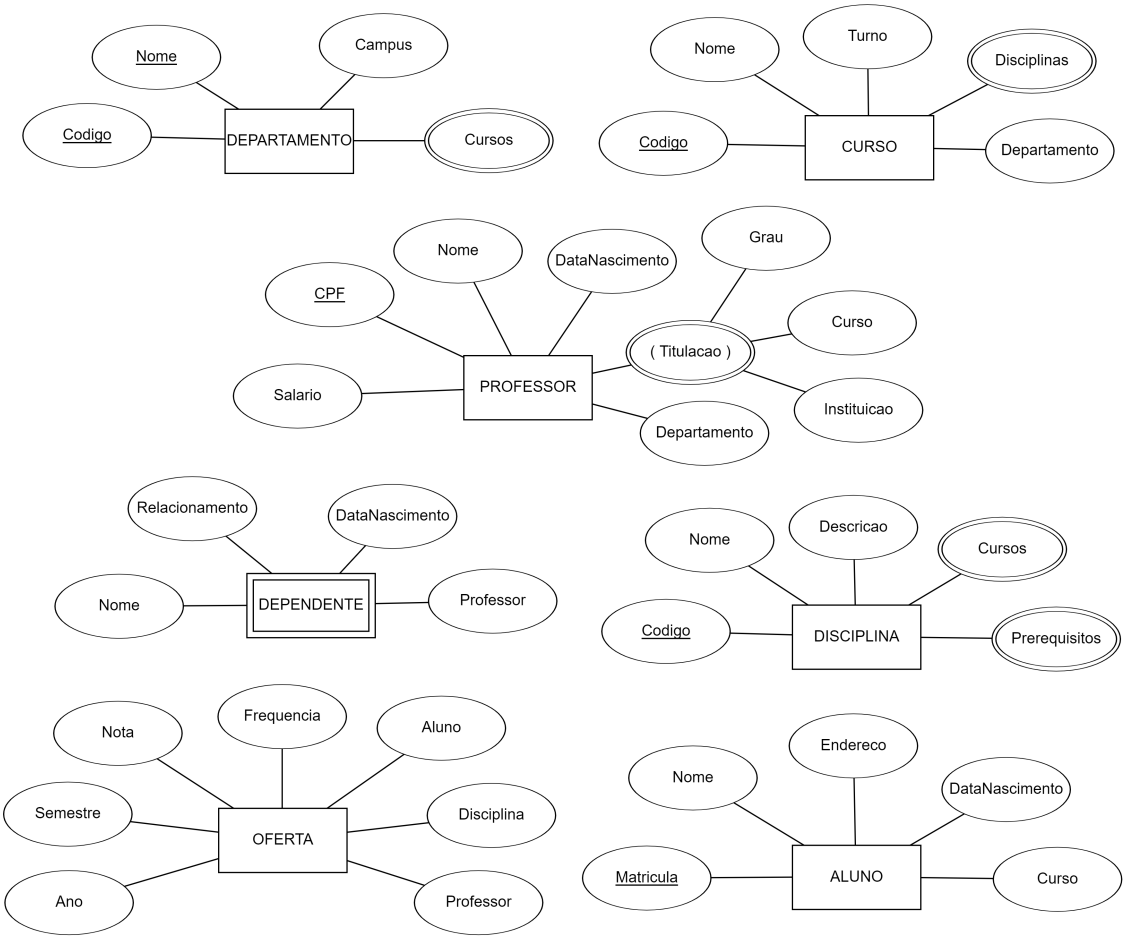
Domínio dos Atributos

Cada atributo de uma entidade é mapeada para um **conjunto de valores**, ou **domínio**, que especifica o conjunto de valores que pode ser atribuído a esse atributo. O atributo salário de um PROFESSOR por exemplo tem como domínio o conjunto de números decimais não-negativos. Já o atributo turno de um curso tem como domínio os caracteres 'V' (vespertino) e 'N' (noturno).

2.2.3 Projeto Conceitual Inicial do Banco de Dados UNIVERSIDADE

Pode-se agora definir no modelo ER as entidades do banco de dados UNIVERSIDADE. A Figura 2.3 define as entidades e os atributos de cada entidade. Na próxima seção será introduzido o conceito de relacionamentos entre as entidades e o modelo será refinado.

Figura 2.3: Diagrama ER inicial para o banco de dados UNIVERSIDADE contendo apenas as entidades e atributos.



2.3 Tipos de Relacionamentos e Restrições Estruturais

Na Figura 2.3 há diversos relacionamentos implícitos entre as entidades. Por exemplo, o atributo chefe do DEPARTAMENTO se relaciona ao professor que chefia o departamento. No modelo ER essas relações não devem ser representadas como atributos, mas sim como **relacionamentos**.

2.3.1 Tipos de Relacionamentos, Conjuntos e Instâncias

Um **tipo de entidade** define uma coleção (ou conjunto) de entidades que têm os mesmos atributos. A coleção de todos os elementos de um tipo de entidade é chamada de **conjunto da entidade**.

Um **tipo de relacionamento** R relaciona n tipos de entidades $E_1, E_2, \ldots, E_n$ e define um **conjunto de relações** entre esses tipos de entidades. Cada relacionamento r_i do tipo de relacionamento R é uma associação de entidades, onde a associação inclui exatamente uma entidade de cada tipo de entidade envolvida [2]. Por exemplo, considere o tipo de relacionamento TRABALHA entre PROFESSOR e DEPARTAMENTO, que relaciona cada professor com o departamento no qual ele trabalha. A coleção de todos os elementos de um tipo de relacionamento é chamada **conjunto do relacionamento**. Nos diagramas ER tipos de relacionamentos são representados como losangos que são conectados por linhas retas às entidades que dele participam.

2.3.2 Graus de Relacionamentos, Papéis e Relacionamentos Recursivos

O grau de um relacionamento é o número de entidades que participam dele. Assim, o relacionamento TRABALHA é de grau dois, pois relaciona um PROFESSOR a um DEPARTAMENTO. Um relacionamento de grau dois é chamado **binário**, enquanto um relacionamento de grau três é chamado **ternário** [2]. Um exemplo de relacionamento ternário é OFERTA, que relaciona uma DISCIPLINA a um PROFESSOR e a um ALUNO. Nesse caso oferta foi representada como uma **entidade associativa** ligando as três entidades que relaciona. Raramente um relacionamento envolve mais de três entidades.

Cada entidade que participa de um relacionamento representa um determinado **papel**. Por exemplo, no relacionamento TRABALHA o PROFESSOR representa o papel de empregado e o departamento representa o papel de empregador. Em alguns casos a mesma entidade participa mais de uma vez num mesmo relacionamento em diferentes papéis. Esses são os **relacionamentos recursivos**. Por exemplo, a entidade DISCIPLINA participa duas vezes do relacionamento PREREQUISITO, uma no papel de "depende de" e outra no papel de "é necessária para".

2.3.3 Restrições em Relacionamentos

Comumente relacionamentos têm restrições que limitam seu número de participantes. Essas restrições são derivadas do mini-mundo que o modelo ER representa. Por exemplo, um PROFESSOR deve ser associado a exatamente um DEPARTAMENTO. Esses tipos de restrições devem ser representadas no modelo. Há dois tipos principais de restrições em relacionamentos: as de taxa de cardinalidade e de participação.

Restrições de Taxa de Cardinalidade

A **taxa de cardinalidade** de um relacionamento representa o número máximo de instâncias de um relacionamento que uma entidade por participar. Por exemplo, para o relacionamento TRABALHA a taxa de cardinalidade DEPARTAMENTO:PROFESSOR é de 1:N representando que cada departamento pode estar relacionado a vários professor (N) enquanto que um professor pode estar relacionado a apenas um departamento. As taxas de cardinalidade possíveis para relacionamentos binários são: 1:1, 1:N, N:1 e M:N.

Um exemplo de relacionamento 1:1 é OFERECE, que associa um departamento ao curso que oferece. O relacionamento CONTEM é um relacionamento M:N pois um curso contém várias disciplinas e uma disciplina pode estar presente em vários cursos.

As restrições de taxa de cardinalidade são representadas na ponta da linha que liga o relacionamento à entidade. Uma ponta com um traço indica participação individual da entidade no relacionamento; uma ponta tripla representa a participação de N elementos da entidade no relacionamento. Por exemplo, na Figura 2.1 participam do relacionamento TRABALHA um DEPARTAMENTO e N entidades de PROFESSOR.

Restrições de Participação

A **restrição de participação** especifica se a existência de uma entidade depende dela ser relacionada a outra via relacionamento. Essa restrição especifica um número mínimo de entidades que deve participar de um relacionamento. Há dois tipos de restrição de participação: total e parcial [2]. Se uma universidade define que cada professor tem que trabalhar para um departamento, esse professor existe apenas se estiver associado a um departamento. Assim, a participação de PROFESSOR no relacionamento TRABALHA é chamada **total**.

Como exemplo, não se espera que cada departamento ofereça um curso, dessa forma a participação de DEPARTAMENTO no relacionamento OFERECE é **parcial**, significando que alguns departamentos são relacionados a um curso via OFERECE, mas não necessariamente todos. Um exemplo dessa situação é o departamento de ciências exatas em um instituto de engenharia. Esse departamento oferece diversas disciplinas mas não é responsável por nenhum dos cursos de engenharia. As restrições de taxa de cardinalidade e de participação são consideradas conjuntamente como as **restrições estruturais** de um relacionamento.

As restrições de participação são representadas também na ponta da linha que conecta o relacionamento à entidade, antes do indicador de taxa de cardinalidade. Uma participação total é representada por um círculo e uma participação parcial é representada por um traço. Por exemplo, considere na Figura 2.1 que há uma participação parcial de DEPARTAMENTO no relacionamento OFERECE e uma participação total de CURSO nesse mesmo relacionamento.

Atributos de Relacionamentos

Relacionamentos também podem ter atributos. Por exemplo, a nota que um aluno obteve em uma OFERTA de disciplina pode ser incluída como um atributo do relacionamento OFERTA. Outro exemplo é a DataInicio em que um professor começou a trabalhar em um departamento, que pode ser definido como um atributo do relacionamento TRABALHA.

2.4 Entidades Fracas

Entidades que não têm atributos chave próprios são chamadas **entidades fracas**. Registros pertencentes a uma entidade fraca são identificadas por serem relacionadas a outras entidades em combinação com um ou mais de seus atributos. Essa outra entidade é chamada **entidade proprietária** e o relacionamento que relaciona a entidade fraca a sua proprietária é chamado **relacionamento identificador** da entidade fraca [2].

Uma entidade fraca sempre tem uma restrição de participação total em seu relacionamento identificador. Por exemplo, a entidade DEPENDENTE, relacionada a um PROFESSOR. Os atributos de um dependente são seu nome, sua data de nascimento e relacionamento (com o professor). É possível que dois dependentes de dois professores distintos tenham o mesmo nome, gênero e relacionamento. Esses são identificados como registros distintos apenas ao examinar o professor com o qual estão relacionados.

Uma entidade fraca normalmente tem uma chave parcial, que é o atributo que pode identificar unicamente registros de uma entidade fraca que são relacionadas à mesma entidade proprietária. Nos diagramas ER, ambas entidade fraca e seu relacionamento identificador são distinguidos por conter linhas duplas no retângulo e losango que os representam [2].

2.5 Refinando o Modelo ER do Banco de Dados UNIVERSIDADE

Agora é possível refinar o modelo ER presente na Figura 2.3 mudando os atributos que representam relacionamentos para relacionamentos explícitos. As restrições estruturais são determinadas de acordo com os requisitos listados na Seção 2.1. Nesse exemplo podem ser especificados os seguintes relacionamentos:

TRABALHA que é um relacionamento 1:N entre DEPARTAMENTO e PROFESSOR. Ambas participações são totais, indicando que todo professor tem que fazer parte de um departamento e todo departamento tem que ter ao menos um professor;

OFERECE um relacionamento 1:1 entre DEPARTAMENTO e CURSO. A participação de CURSO é total, indicando que todo CURSO tem que ter um DEPARTAMENTO responsável, enquanto que a participação de DEPARTAMENTO foi considerada parcial, permitindo que exista departamento que não seja responsável por nenhum curso.

LECIONA um relacionamento 1:N entre PROFESSOR e OFERTA. A participação do PROFESSOR é total, indicando que toda OFERTA tem que ter um professor e a participação de OFERTA é parcial pois pode haver professor que não OFERTA quaisquer disciplina;

OCORRE um relacionamento 1:N entre DISCIPLINA e OFERTA. A participação da DISCIPLINA é total, indicando que toda OFERTA tem que ter uma disciplina e a participação de OFERTA é parcial pois pode haver disciplina que não seja ofertada;

MATRICULA um relacionamento 1:N entre ALUNO e OFERTA. A participação do ALUNO é total, indicando que toda OFERTA tem que ter um aluno e a participação de OFERTA é parcial pois pode haver aluno que não se matricula em nenhuma OFERTA;

CONTEM um relacionamento M:N entre DISCIPLINA e CURSO, indicando que uma disciplina pode estar presente em diversos cursos e um curso é composto de diversas disciplinas. Ambas participações são totais;

CURSA um relacionamento 1:N entre ALUNO e CURSO, indicando que um aluno cursa apenas um curso e que um curso possui vários alunos. Ambas participações são totais, indicando que não há ALUNO sem CURSO, nem CURSO sem ALUNO;

PREREQUISITO um relacionamento M:N entre DISCIPLINA (no papel de "depende de") e DISCIPLINA (no papel de "é necessária para"). Ambas participações são determinadas como parciais, visto que não há restrição que obriga uma disciplina a ter pré-requisito nem uma disciplina a ser pré-requisito de outra;

DEPENDE um relacionamento 1:N entre PROFESSOR e DEPENDENTE, que também é o relacionamento identificador da entidade fraca DEPENDENTE. A participação de PROFESSOR é parcial, enquanto que a participação de DEPENDENTE é total.

2.6 Exercícios

Exercicio 2.1 Defina os seguintes termos: entidade, atributo, relacionamento, atributo composto, atributo multi-valorado, atributo derivado, atributo-chave, domínio de um atributo. ■

Exercicio 2.2 Apresente outros exemplos de contextos em que atributos compostos, entidades fracas, relacionamentos recursivos e relacionamentos ternários podem ser aplicados. ■

Exercicio 2.3 Crie um modelo ER para o banco de dados EMPRESA com as necessidades de informação descritas a seguir. O banco de dados EMPRESA armazena informações sobre seus empregados, departamentos e projetos. Suponha que após a coleta de requisitos os projetistas do banco de dados proveram a seguinte descrição do mini-mundo a ser representado:

- A empresa é organizada em departamentos. Cada departamento tem um nome único, um número único e um empregado em particular que o gerencia. Deve-se armazenar a data

de início em que o empregado começou a gerenciar o departamento. Um departamento pode ter várias localizações.
- Um departamento controla vários projetos, cada qual com seu nome único, número único e localização única.
- O banco de dados deve armazenar o nome, CPF, endereço, salário, gênero e data de nascimento dos empregados. Um empregado é associado a um departamento, mas pode trabalhar em vários projetos, que não são necessariamente controlados pelo mesmo departamento. É necessário manter os dados das horas por semana que um empregado trabalha em um dado projeto, bem como o supervisor direto de cada empregado (que é também um empregado).
- O BD deve manter os dados dos dependentes de cada empregado, incluindo o nome, gênero, data de nascimento e o relacionamento do dependente com o empregado.

■

3. Mapeamento Modelo ER-Relacional

Esse capítulo mostra como um modelo entidade-relacionamento é mapeado para um banco de dados relacional. Há várias ferramentas CASE (*Computer Aided Software Engineering*) que manipulam modelos ER e podem gerar um BD relacional automaticamente, como o ERDPlus [3] e o MySQL Workbench [4].

3.1 Algoritmo de Mapeamento ER-Relacional

A Figura 3.1 apresenta o esquema de banco de dados relacional gerado a partir do modelo ER presente na Figura 2.1. Os passos para a geração desse esquema serão apresentados nessa seção. Há formas alternativas para realizar alguns dos passos descritos a seguir, mas por simplicidade apenas a abordagem mais convencional será descrita.

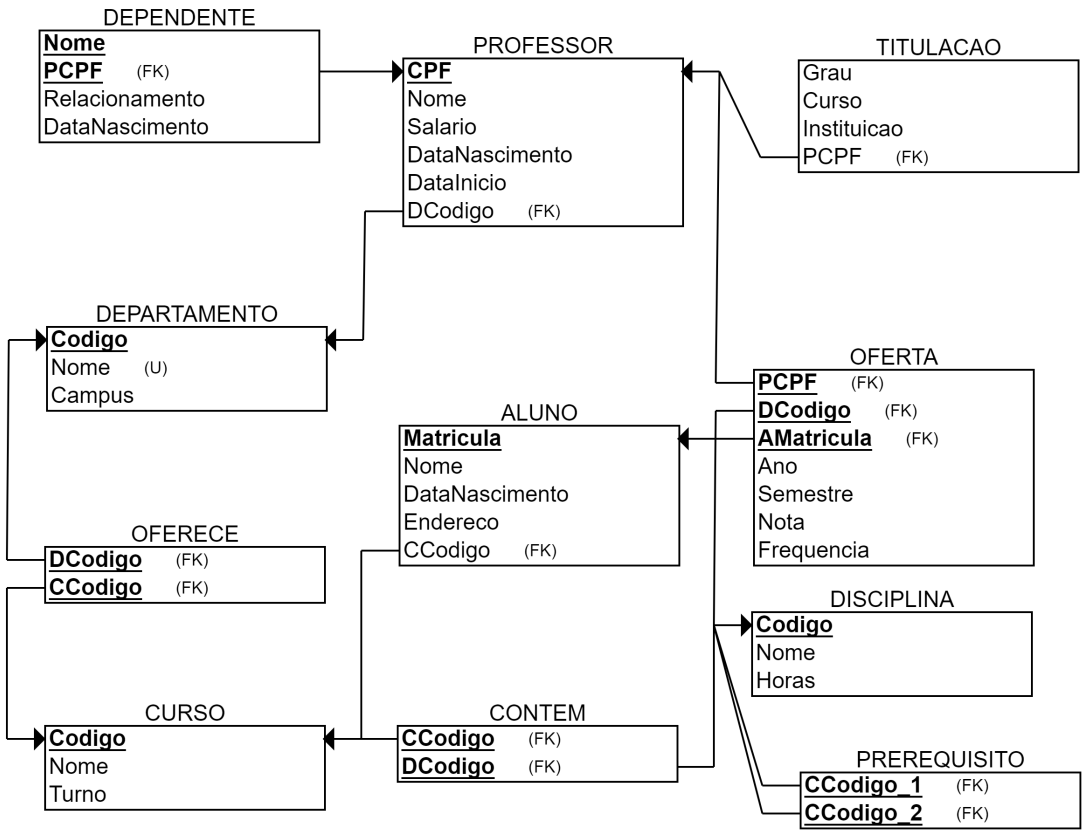
3.1.1 Mapeamento de Tipos de Entidade Regulares

Para cada tipo de entidade forte E no modelo ER, crie uma relação R que inclui todos os atributos de E. Crie um atributo para cada componente de um atributo composto. Escolha um dos atributos chave da entidade para ser a chave primária de R. Se múltiplas chaves forem especificadas para E, a informação de cada chave é mantida para especificar chaves adicionais únicas. Foram criadas relações para os tipos de entidades DEPARTAMENTO, PROFESSOR, CURSO, ALUNO, OFERTA e DISCIPLINA.

3.1.2 Mapeamento de Tipos de Entidades Fracas

Para cada tipo de entidade fraca W no modelo ER com entidade proprietária E, crie uma relação R que inclui todos os atributos de W. Adicionalmente, inclua a chave primária da entidade proprietária como uma chave estrangeira em R. A chave primária de R será a combinação da chave estrangeira da entidade proprietária e sua chave parcial. Foi criada uma relação para tipo de entidade fraca DEPENDENTE, contendo o CPF do PROFESSOR como uma chave estrangeira. A chave primária da relação criada é composta da união no Nome do DEPENDENTE e do CPF do PROFESSOR.

Figura 3.1: Esquema relacional do banco de dados UNIVERSIDADE.



3.1.3 Mapeamento de Tipos de Relacionamentos Binários 1:1

Para cada tipo de relacionamento binário 1:1 R no esquema ER, identificar os tipos de entidades E_1 e E_2 que participam de R. Escolha uma das entidades E_1 e inclua a chave primária de E_2 como chave estrangeira em E_1. É melhor escolher a entidade com participação total em R para o papel de E_1. Se houver, inclua os atributos de R em E_1. O tipo de relacionamento CHEFIA é 1:1, nesse caso, a relação DEPARTAMENTO recebeu a chave primária de PROFESSOR como chave estrangeira e o atributo DataInicio foi também incluído na relação DEPARTAMENTO.

3.1.4 Mapeamento de Tipos de Relacionamentos Binários 1:N

Para cada tipo de relacionamento binário 1:N R, identifique a relação R_1 que representa o tipo de entidade participante do lado N do tipo de relacionamento. Inclua como chave estrangeira em R_1 a chave primária de R_2. Inclua qualquer atributo de R, se houver, na relação R_1. No modelo ER da Figura 2.1 foram identificados os seguinte tipos de relacionamento 1:N: TRABALHA, LECIONA, MATRICULA, OCORRE, CURSA e DEPENDE.

A chave primária de DEPARTAMENTO foi incluída como chave estrangeira em PROFESSOR; a chave primária de PROFESSOR foi incluída como chave estrangeira em OFERTA; a chave primária de ALUNO foi incluída como chave estrangeira em OFERTA; a chave primária de DISCIPLINA foi incluída como chave estrangeira em OFERTA; a chave primária de CURSO foi incluída como chave estrangeira em ALUNO; para o tipo de relacionamento DEPENDE a chave primária de PROFESSOR já havia sido incluída como chave estrangeira em DEPENDENTE no passo da Seção 3.1.2.

3.1.5 Mapeamento de Tipos de Relacionamentos Binários M:N

Para cada tipo de relacionamento binário M:N R, crie uma nova relação para representar R. Inclua como chave estrangeira a chave primária de ambas relações R_1 e R_2 que fazem parte do tipo de relacionamento. A chave primária da entidade resultante será composta da combinação das chaves estrangeiras de R_1 e R_2. Os atributos de R, se houver, também serão incluídos nessa nova relação. No esquema ER da Figura 2.1 foram identificados os tipos de relacionamento M:N CONTEM e CURSA. Foram criadas as relações CONTEM e CURSA com as chaves estrangeiras de dos tipos de entidade que participam de cada relação.

3.1.6 Mapeamento de Atributos Multi-Valorados

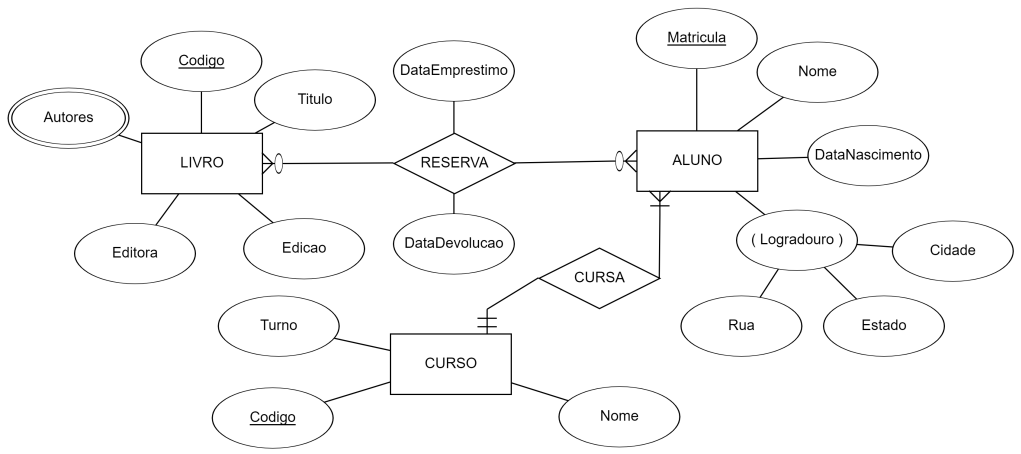
Para cada atributo multi-valorado A, crie uma nova relação R. Essa relação irá conter um atributo relacionado a A e a chave primária do tipo de entidade à qual pertence como chave estrangeira. Caso A seja composto, os atributos de A também farão para da relação R. No esquema ER da Figura 2.1, o atributo Titulacao de PROFESSOR gerou a relação TITULACAO que contém o CPF do PROFESSOR como chave estrangeira e os (sub-)atributos de Titulacao.

3.1.7 Mapeamento de Tipos de Relacionamentos N-ários

Para cada tipo de relacionamento N-ário, crie uma relação R que o representa. A chave primária de cada tipo de entidade que participa do relacionamento deve ser incluída como uma chave estrangeira em R. A chave primária de R seja a combinação das chaves estrangeiras dos tipos de entidade que participam do relacionamento. Caso haja atributos relacionados ao tipo de relacionamento, eles serão também incluídos como atributos na relação R. No modelo ER da UNIVERSIDADE o tipo de relacionamento OFERTA é ternário. O tipo de relacionamento já foi representado como uma entidade associativa. A relação oferta terá então como chaves estrangeiras, o CPF do PROFESSOR, a matrícula do ALUNO e o código da DISCIPLINA, além dos atributos Ano, Semestre e Nota.

3.2 Exercícios

Exercicio 3.1 Mapeie o modelo entidade-relacionamento da figura abaixo, que representa o banco de dados BIBLIOTECA, para o modelo relacional.



■

4. SQL Básica

Structured Query Language (SQL) é a linguagem padrão para manipular bancos de dados relacionais. A SQL foi desenvolvida nos anos 70 pela IBM e foi adotada por várias outras companhias que desenvolvem SGBDs [1].

4.1 Definição e Tipos de Dados em SQL

SQL usa os termos **tabela**, **linha** e **coluna** para os termos relacionais relação, tupla e atributo, respectivamente. O principal comando SQL para definição de dados é o CREATE, que pode ser usado para criar esquemas, tabelas, tipos dentre outros.

O comando CREATE TABLE é usado para criar uma nova relação, seguido do nome da relação e entre parênteses os atributos dessa relação. Para cada atributo é necessário especificar o seu tipo de dados, que pode ser INT (número inteiro), FLOAT (número decimal), CHAR (caractere) VARCHAR(N) (sequencia com N caracteres), DATE (data), dentre outros menos usados. Para cada atributo é possível ainda adicionar a declaração NOT NULL indicando que não pode haver tuplas dessa relação sem valor para o dado atributo. Na criação da tabela é ainda necessário especificar qual é a chave primária, com a declaração PRIMARY KEY e, se houver, a(s) chave(s) estrangeira(s) com a declaração FOREIGN KEY indicando via REFERENCES a tabela e atributo ao qual se referencia. O comando SQL a seguir cria a tabela ALUNO do banco de dados UNIVERSIDADE.

```
CREATE TABLE ALUNO
(
    Matricula          INT             NOT NULL,
    Nome               VARCHAR(45)     NOT NULL,
    DataNascimento     DATE            NOT NULL,
    Endereco           VARCHAR(45)     NOT NULL,
    CCodigo            INT             NOT NULL,
    PRIMARY KEY (Matricula),
    FOREIGN KEY (CCodigo) REFERENCES CURSO(Codigo)
);
```

Os comandos DROP e ALTER são responsáveis por excluir e alterar uma tabela respectivamente. O exemplo abaixo exclui a tabela ALUNO e adiciona uma coluna (atributo) na tabela PROFESSOR. Esses comandos não serão abordados em detalhes pois ferramentas CASE apresentam a funcionalidade de gerar e executar esses comandos de forma intuitiva via interfaces gráficas.

```
DROP TABLE ALUNO;

ALTER TABLE    PROFESSOR
      ADD      COLUMN 'Endereco' VARCHAR(45) NOT NULL AFTER 'DCodigo';
```

4.2 Consultas SQL

Consultas em SQL podem ser bem complexas. Consultas mais simples serão apresentadas e gradualmente consultas mais complexas serão introduzidas. A forma básica da declaração SELECT é formada pelas três cláusulas SELECT, FROM, WHERE e tem a seguinte forma:

```
SELECT  <lista de atributos>
FROM    <lista de tabelas>
WHERE   <condicoes>;
```

Em SQL, os operadores lógicos básicos de comparação são =, <, <=, >, >= e <>. Eles correspondem respectivamente aos operadores de álgebra relacional =, <, $\leq$, >, $\geq$ e $\neq$. A declaração de consulta básica SELECT será apresentada tendo como exemplo algumas consutlas simples ao esquema construído ao longo dessa apostila e representado na Figura 3.1.

CONSULTA 1. Recuperar a data de nascimento e salário dos professores cujo nome é 'George Fonseca'.

```
SELECT  DataNascimento, Salario
FROM    PROFESSOR
WHERE   Nome='George Fonseca';
```

Essa consulta envolve a tabela PROFESSOR listada na cláusula FROM. A consulta seleciona as tuplas de PROFESSOR que satisfazem a condição na cláusula WHERE e então projeta o resultado nos atributos DataNascimento e Salario listados na cláusula SELECT.

CONSULTA 2. Recuperar o CPF e o salário de todos os professores que trabalham nos departamento cujo campus é 'Ipatinga'.

```
SELECT  CPF, Salario
FROM    PROFESSOR, DEPARTAMENTO
WHERE   Campus='Ipatinga' AND Codigo=DCodigo;
```

Na cláusula WHERE da Consulta 2 a condição Campus='Ouro Preto' é uma **condição de seleção** que seleciona a tupla particular de interesse da tabela DEPARTAMENTO, pois Campus é um atributo de DEPARTAMENTO. A condição Codigo=DCodigo é chamada **condição de junção**, pois combina duas tuplas: uma de DEPARTAMENTO e outra de PROFESSOR sempre que o valor de Codigo em DEPARTAMENTO é igual ao valor de DCodigo em PROFESSOR. De modo geral, qualquer número de seleções e condições de junção podem ser especificados em uma consulta SQL.

4.2.1 Nomes de Atributos Ambíguos, Pseudônimo e Variáveis de Tupla

Na SQL o mesmo nome pode ser usado para mais de um atributo em diferentes tabelas. No esquema da Figura 3.1 por exemplo, o atributo "Nome" está presente nas tabelas DEPARTAMENTO, PROFESSOR, CURSO e ALUNO. Se esse for o caso e uma consulta multi-tabela se refere a dois ou mais atributos com o mesmo nome, devemos qualificar o nome do atributo de acordo com o nome da tabela para prevenir ambiguidade. Isso é feito prefixando o nome da tabela ao nome do atributo, separando-os por ponto.

```
SELECT   PROFESSOR.CPF, PROFESSOR.Salario
FROM     PROFESSOR, DEPARTAMENTO
WHERE    DEPARTAMENTO.Campus='Ipatinga' AND
         DEPRATAMENTO.Codigo=PROFESSOR.DCodigo;
```

É possível ainda renomear nomes de tabelas para nomes mais curtos criando um pseudônimo, como no exemplo da Consulta 3. De fato essa prática é recomendável para facilitar a legibilidade das consultas SQL.

CONSULTA 3. Recupere o código e o nome das disciplinas que pertencem ao curso de 'Medicina'.

```
SELECT   D.Codigo, D.Nome
FROM     DISCIPLINA AS D, CONTEM AS C, CURSO AS CS
WHERE    D.Codigo=C.DCodigo AND CS.Codigo=C.CCodigo
         AND C.Nome='Medicina';
```

4.2.2 Cláusura WHERE Não Especificada e Asterisco

Uma cláusula WHERE faltando significa que não há condição na seleção das tuplas, assim, todas as tuplas especificadas na cláusula FROM serão selecionadas.

CONSULTA 4. Selecione os CPFs de todos os professores.

```
SELECT   CPF
FROM     PROFESSOR;
```

CONSULTA 5. Selecione os CPFs de todos os empregados e todas as combinações de CPFs de professores e nomes de departamentos.

```
SELECT   E.CPF, D.Nome
FROM     EMPREGADO AS E, DEPARTAMENTO AS D;
```

Para recuperar os valores de todos os atributos das tuplas selecionadas não é necessário listar todos os atributos na SQL, é possível especificar apenas um asterisco (*).

CONSULTA 6. Recuperar todos os atributos dos professores que trabalham no DEPARTAMENTO cujo código é 'DECOM'.

```
SELECT   *
FROM     PROFESSOR
WHERE    DCodigo='DECOM'
```

CONSULTA 7. Recuperar todos os atributos dos professores e os atributos do DEPARTAMENTO em que eles trabalham para cada empregado do departamento 'Computacao'.

```
SELECT   P.*
FROM     PROFESSOR AS P, DEPARTAMENTO AS D
WHERE    P.DCodigo=D.Codigo AND D.Nome='Computacao'
```

4.2.3 Tabelas como Conjuntos em SQL

SQL usualmente não trata uma tabela como um conjunto, e sim como um multi-conjunto, assim tuplas duplicadas podem aparecer como resultado de uma consulta. Se deseja-se eliminar duplicatas dos resultados de uma consulta SQL a palavra-chave DISTINCT deve ser usada na cláusula SELECT, especificando que apenas tuplas distintas devem permanecer no resultado.

CONSULTA 8. Recuperar todos os salários distintos de todos os professores.

```
SELECT   DISTINCT Salario
FROM     PROFESSOR
```

A SQL incorporou diretamente algumas das operações da teoria dos conjuntos, como operações de união (UNION), diferença (DIFFERENCE) e interseção (INTERSECT). Os resultados dessas operações São conjuntos de tuplas, assim, duplicatas são eliminadas do resultado. As operações sobre conjuntos se aplicam apenas a tabelas com tipos compatíveis, ou seja, as tabelas para as quais a operação está sendo processada têm que ter os mesmos atributos e na mesma ordem.

CONSULTA 9. Listar todos o nome de todas as pessoas (professores e alunos) que participaram da oferta de disciplinas no ano de 2020.

```
(SELECT   P.Nome
 FROM     PROFESSOR AS P, OFERTA AS O
 WHERE    P.CPF=O.PCPF AND O.Ano=2020)
UNION
(SELECT   A.Nome
 FROM     ALUNO AS A, OFERTA AS O
 WHERE    A.Matricula=O.AMatricula AND O.Ano=2020);
```

A SQL também permite as operações multi-conjuntos correspondentes, que são seguidas pela palavra-chave ALL (UNION ALL, DIFFERENCE ALL, INTERSECT ALL). Esses resultados são multi-conjuntos (duplicatas não são eliminadas).

4.2.4 Correspondência de Padrão de Substring e Operações Aritméticas

Uma funcionalidade da SQL permite a comparação de apenas partes de caracteres de uma string, usando o operador de comparação LIKE. Isso pode ser usado para **correspondência de padrões de strings**. Strings parciais são especificadas usando dois caracteres especiais: % substitui um número arbitrário de zero ou mais caracteres e sublinhado (_) substitui um único caractere. Por exemplo, considere a seguinte consulta.

CONSULTA 10. Recuperar todos alunos cujo endereço contém 'Ipatinga'.

```
SELECT  Nome
FROM    ALUNO
WHERE   Endereco LIKE '%Ipatinga%';
```

Para recuperar todos os professores que nasceram nos anos 70 podemos especificar que o nono caractere da data de nascimento (no formato 'dd/mm/aaaa') seja 5.

CONSULTA 11. Recuperar todos os professores nascidos nos anos 70.

```
SELECT  Nome
FROM    PROFESSOR
WHERE   DataNascimento LIKE '________5_';
```

Outra característica permite o uso de operações aritméticas nas consultas. Os operadores aritméticos padrão para adição (+), subtração (-), multiplicação (*) e divisão (/) podem ser aplicados a valores numéricos ou atributos com domínios numéricos. Por exemplo, suponha que queira-se ver o efeito de dar um aumento de 10% para todos os professores que trabalham no departamento 'Computacao'.

CONSULTA 12. Mostrar os salários resultantes se cada professor que do deprtamento 'Computacao' recebesse um aumento de 10%.

```
SELECT  P.Nome, P.Salario AS SalAumentado
FROM    PROFESSOR AS P, DEPARTAMENTO AS D
WHERE   D.Codigo=P.DCodigo AND D.Nome='Computacao';
```

4.2.5 Ordenando os Resultados de uma Consulta

A SQL permite ao usuário ordenar as tuplas resultantes de uma consulta pelos valores de um ou mais dos atributos que aparecem no resultado da consulta usando a cláusula ORDER BY. A Consulta 13 apresenta um exemplo.

CONSULTA 13. Recuperar a lista dos professores e dos departamentos nos quais eles estão trabalham ordenados pelo nome do departamento e, dentro de cada departamento, ordenar alfabeticamente pelo nome.

```
SELECT    D.Nome, P.Nome
FROM      DEPARTAMENTO AS D, PROFESSOR AS P
WHERE     D.Codigo=P.DCodigo
ORDER BY  D.Nome, P.Nome;
```

A ordem padrão é a ordem ascendente dos valores. Pode-se especificar a palavra-chave DESC se deseja-se ver os resultados na ordem decrescente de valores. Por exemplo, caso se desejasse os nomes dos departamentos em ordem decrescente a cláusula ORDER BY ficaria:

```
ORDER BY D.Nome DESC, P.Nome;
```

4.2.6 Funções de Agregação em SQL

Funções de agregação são usadas para sumarizar informações de múltiplas tuplas em uma única-tupla sumário. Várias funções de agregação existem como a contagem de elementos (COUNT), o mínimo (MIN) e máximo (MAX) de uma série de valores, a soma (SUM) e a média (AVG) de um conjunto de dados. Há ainda funções de agregação para cálculos estatísticos mais avançados, porém estas não serão abordadas aqui. Essas funções podem ser usadas em uma cláusula SELECT. Essas funções serão ilustradas através das consultas a seguir.

CONSULTA 14. Recuperar a soma dos salários de todos os professores, o salário máximo, o salário mínimo e o salário médio.

```
SELECT    SUM (Salario), MIN (Salario), MAX (Salario), AVG (Salario)
FROM      PROFESSOR;
```

Pode-se usar a palavra-chave AS para criar renomear os nomes das colunas resultantes da consulta:

```
SELECT   SUM (Salario) AS TotalSal, MIN (Salario) AS MinSal,
         MAX (Salario) AS MaxSal, AVG (Salario) AS MedSal
FROM     PROFESSOR;
```

CONSULTA 15. Recuperar o número total de professores que trabalham no departamento 'Computacao'.

```
SELECT   COUNT (*)
FROM     PROFESSOR AS P, DEPARTAMENTO AS D
WHERE    D.Codigo=P.DCodigo AND D.Nome='Computacao';
```

4.2.7 Agrupamento: Cláusulas GROUP BY e HAVING

Muitas vezes deseja-se aplicar funções de agregação a subgrupos de tuplas em uma relação, onde esses subgrupos são baseados em alguns valores de atributos. Por exemplo, se desejarmos encontrar o salário médio dos empregados em cada departamento. Nesses casos precisa-se particionar as tabelas em conjuntos sem sobreposição (ou grupos) de tuplas. Cada grupo consistirá das tuplas que têm o mesmo valor para um determinado **atributo de agrupamento**. A SQL tem a cláusula GROUP BY para essa finalidade. A cláusula GROUP BY especifica os atributos de agrupamento, que devem também aparecer na cláusula SELECT.

CONSULTA 16. Para cada departamento, recuperar o código do departamento, o número de professores no departamento e seu salário médio.

```
SELECT    DNumero, COUNT (*), AVG (Salario)
FROM      PROFESSOR
GROUP BY  DNo
```

Algumas vezes deseja-se recuperar valores para funções de agregação apenas para grupos que satisfazem determinada condição. Isso pode ser feito pela cláusula HAVING, como o exemplo da consulta a seguir.

CONSULTA 17. Para cada departamento, no qual mais de dois professores trabalham, recuperar o

código do departamento, o nome do departamento e o número de professores que nele trabalham.

```
SELECT    D.Codigo, D.Nome, COUNT (*)
FROM      PROFESSOR, DEPRTAMENTO
WHERE     D.Codigo=P.DCodigo
GROUP BY D.Codigo, D.Nome
HAVING    COUNT (*) > 2;
```

4.3 Declarações INSERT, DELETE e UPDATE

Na SQL três comandos

O comando INSERT adiciona uma tupla a uma tabela. Deve-se especificar o nome da relação e um lista de valores para a tupla. Esses valores deve estar na mesma ordem em que os atributos foram especificados no comando CREATE TABLE. O exemplo a seguir adiciona a aluna Mariana Gomes à tabela ALUNO.

```
INSERT INTO ALUNO VALUES
(
    '2018005',
    'Mariana Gomes',
    '2001-02-28',
    'Rua Pedro A. Cabral, Belo Horizonte - MG',
    '1'
);
```

O comando DELETE remove uma tupla de uma tabela. Ele inclui uma cláusula WHERE, de forma similar às consultas, para selecionar as tuplas a serem deletadas. O exemplo a seguir deleta o ALUNO de nome 'Caio Nunes' da tabela ALUNO.

```
DELETE FROM   ALUNO
WHERE         Nome='Caio Nunes';
```

O comando UPDATE é usado para modificar os dados de uma ou mais tuplas de uma tabela. Assim como no comando DELETE, uma cláusula WHERE deve ser especificada para definir quais tuplas sofrerão atualização. A SQL a seguir atualiza a data de nascimento da aluna 'Ana Pedrosa'.

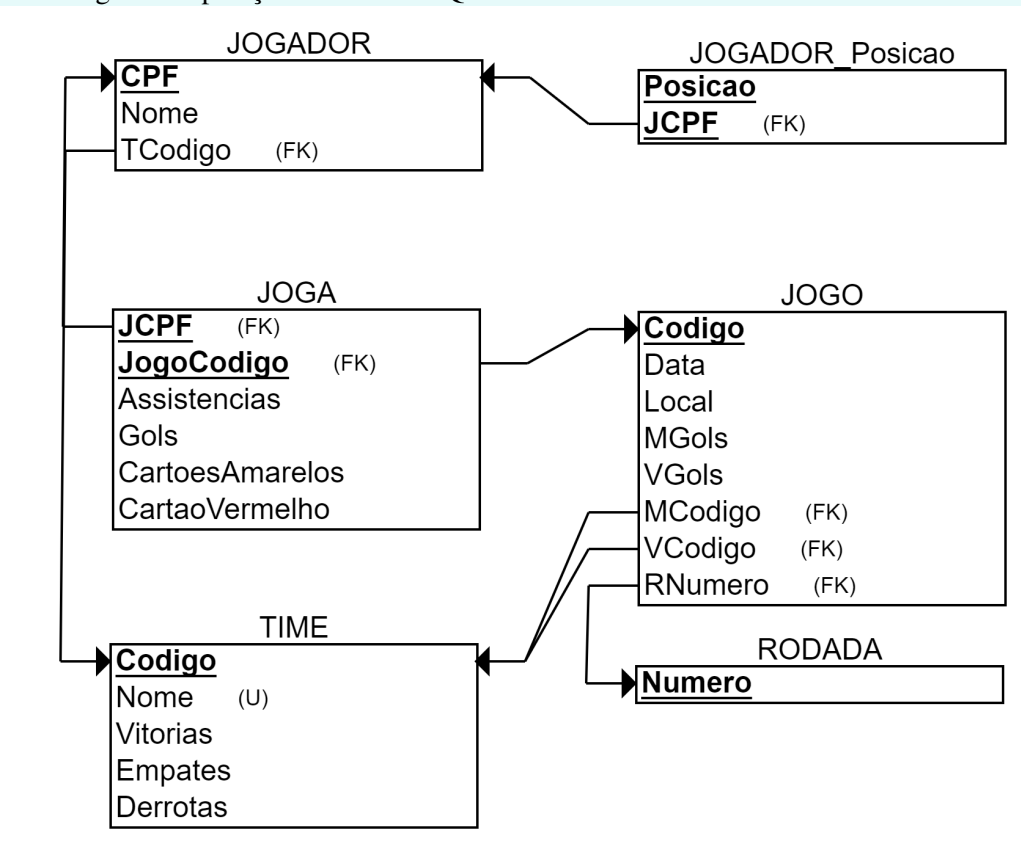
```
UPDATE    ALUNO
SET       Endereco='Av. Ipanema, Itabira - MG'
WHERE     Nome='Ana Pedrosa';
```

Múltiplas tuplas podem ser atualizadas ao mesmo tempo em um único comando UPDATE, como no exemplo abaixo, que aumenta o salário dos professores do departamento de código '3' em 5%.

```
UPDATE    PROFESSOR
SET       Salario=Salario*1.05
WHERE     DCodigo='3';
```

4.4 Exercícios

Exercicio 4.1 Considerando o modelo relacional do banco de dados CAMPEONATO abaixo, realize as seguintes operações/consultas SQL:



- Adicionar a equipe 'Porto F. C.', de código 1053 à tabela TIME com 0 vitórias, 0 derrotas e 0 empates.
- Adicionar a posição 'Lateral direito' ao jogador de CPF '10120422140'.
- Exibir o código, o nome e o número de vitórias dos times com mais que 5 vitórias.
- Recuperar o nome e o CPF de todos os jogadores que atuam na posição 'Goleiro'.
- Recuperar os dados do jogador que mais fez gols no campeonato (artilheiro).
- Recuperar os times ordenados decrescentemente pelo número de vitórias, e pelo número de empates caso o número de vitórias seja igual.
- Recuperar o número de gols marcados e o número de gols sofridos pela equipe 'Sporting F. C.'.
- Recuperar a equipe que tem o maior número de derrotas.
- Recuperar a média dos salários dos jogadores da equipe 'River F. C.'
- Recuperar o nome dos jogadores que atuaram pela equipe 'América F. C.' na rodada número 3.
- Recuperar o nome das equipes que venceram na rodada 7.

■

Bibliografia

Livros

[1] Christopher John Date. *An introduction to database systems*. Pearson Education India, 1981 (ver página 27).

[2] Ramez Elmasri e Sham Navathe. *Fundamentals of Database Systems*. 7ª edição. Pearson, 2016 (ver páginas 7, 8, 10, 16, 17, 19, 20).

[3] Nenad Jukic, Susan Vrbsky e Svetlozar Nestorov. *Database systems: Introduction to databases and data warehouses*. Pearson, 2014 (ver página 23).

[4] Michael McLaughlin. *MySQL Workbench: Data Modeling Development*. McGraw Hill Professional, 2013 (ver página 23).