capítulo 1

Bancos de dados e usuários de banco de dados

Bancos de dados e sistemas de banco de dados são um componente essencial da vida na sociedade moderna; a maioria de nós encontra diariamente diversas atividades que envolvem alguma interação com um banco de dados. Por exemplo, quando vamos ao banco para depositar ou retirar fundos, fazemos uma reserva de hotel ou de voo, acessamos o catálogo de uma biblioteca virtual para procurar uma referência bibliográfica, ou compramos algo on-line — como um livro, um brinquedo ou um computador —, provavelmente essas atividades envolverão alguém ou algum programa de computador que acessa um banco de dados. Até mesmo a compra de produtos em um supermercado atualiza automaticamente o banco de dados que mantém o controle de estoque dos itens.

Essas interações são exemplos do que podemos chamar de **aplicações de banco de dados tradicionais**, em que a maior parte da informação armazenada e acessada é textual ou numérica. Nos últimos anos, os avanços na tecnologia levaram a interessantes novas aplicações dos sistemas de banco de dados. A nova tecnologia de mídia tornou possível armazenar imagens, clipes de áudio e streams de vídeo digitalmente. Esses tipos de arquivo estão se tornando um componente importante dos **bancos de dados de multimídia**. Os **sistemas de informações geográficas** (**GIS** — Geographic Information Systems) podem armazenar e analisar mapas, dados sobre o clima e imagens de satélite. Sistemas de **data warehousing** e de **processamento analítico on-line** (**OLAP** — On-Line Analytical Processing) são usados em muitas empresas para extrair e analisar informações comerciais úteis de bancos de dados muito grandes, para ajudar na tomada de decisão. A **tecnologia de tempo real** e **banco de dados ativo** é usada para controlar processos industriais e de manufatura. Além disso, técnicas de pesquisa de banco de dados estão sendo aplicadas à World Wide Web para melhorar a busca por informações necessárias feita pelos usuários que utilizam a Internet.

No entanto, para entender os fundamentos da tecnologia de banco de dados, devemos começar das aplicações básicas de banco de dados tradicional. Na Seção 1.1, começamos definindo um banco de dados, e depois explicamos outros termos básicos. Na Seção 1.2, oferecemos um simples exemplo de banco de dados UNIVERSIDADE para ilustrar nossa discussão. A Seção 1.3 descreve algumas das principais características dos sistemas de banco de dados, e as seções 1.4 e 1.5 classificam os tipos de pessoas cujas funções envolvem o uso e a interação com sistemas de banco de dados. As seções 1.6, 1.7 e 1.8 oferecem uma discussão mais profunda sobre as diversas capacidades oferecidas pelos sistemas de banco de dados e discutem algumas aplicações típicas. No final do capítulo é apresentado um resumo.

O leitor que quiser uma introdução rápida aos sistemas de banco de dados pode estudar as seções 1.1 a 1.5, depois pular ou folhear as seções 1.6 a 1.8 e seguir para o Capítulo 2.

1.1 Introdução

Os bancos de dados e sua tecnologia têm um impacto importante sobre o uso crescente dos computadores. É correto afirmar que os bancos de dados desempenham um papel crítico em quase todas as áreas em que os computadores são usados, incluindo negócios, comércio eletrônico, engenharia, medicina, genética, direito, educação e biblioteconomia. O termo *banco de dados* (do original *database*) é tão utilizado que precisamos começar por sua definição. E nossa definição inicial é bastante genérica.

Um **banco de dados** é uma coleção de dados[1] relacionados. Com **dados**, queremos dizer fatos conhecidos que podem ser registrados e possuem significado implícito. Por exemplo, considere os nomes, números de telefone e endereços das pessoas que você conhece. Você pode ter registrado esses dados em uma agenda ou, talvez, os tenha armazenado em um disco rígido, usando um computador pessoal e um software como Microsoft Access ou Excel. Essa coleção de dados relacionados, com um significado implícito, é um banco de dados.

Essa definição de banco de dados é bastante genérica; por exemplo, a coleção de palavras que compõem esta página de texto pode ser considerada dados relacionados e, portanto, constitui um banco de dados. Porém, o uso comum do termo *banco de dados* normalmente é mais restrito e tem as seguintes propriedades implícitas:

- Um banco de dados representa algum aspecto do mundo real, às vezes chamado de **minimundo** ou de **universo de discurso** (**UoD** — Universe of Discourse). As mudanças no minimundo são refletidas no banco de dados.
- Um banco de dados é uma coleção logicamente coerente de dados com algum significado inerente. Uma variedade aleatória de dados não pode ser corretamente chamada de banco de dados.
- Um banco de dados é projetado, construído e populado com dados para uma finalidade específica. Ele possui um grupo definido de usuários e algumas aplicações previamente concebidas nas quais esses usuários estão interessados.

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

Um banco de dados pode ter qualquer tamanho e complexidade. Por exemplo, a lista de nomes e endereços referenciados anteriormente pode consistir em apenas algumas centenas de registros, cada um com uma estrutura simples. Por sua vez, o catálogo computadorizado de uma grande biblioteca pode conter meio milhão de entradas organizadas sob diferentes categorias — por sobrenome do autor principal, por assunto, por título do livro —, com cada uma das categorias organizada alfabeticamente. Um banco de dados de tamanho e complexidade ainda maior é mantido pela Receita Federal para monitorar formulários de imposto de renda preenchidos pelos contribuintes. Se considerarmos que existem 100 milhões de contribuintes e que cada um deles preenche uma média de cinco formulários com aproximadamente 400 caracteres cada um, teríamos um banco de dados de $100 \times 10^6 \times 400 \times 5$ caracteres (bytes) de informação. Se a Receita Federal mantém o registro dos três últimos anos de cada contribuinte, além do ano atual, teríamos um banco de dados de 8×10^{11} bytes (800 gigabytes). Essa imensa quantidade de informações precisa ser organizada e gerenciada de modo que os usuários possam consultar, recuperar e atualizar os dados quando necessário.

Um exemplo de um grande banco de dados comercial é a Amazon.com. Ela contém dados de mais de 20 milhões de livros, CDs, vídeos, DVDs, jogos, eletrônicos, roupas e outros itens. O banco de dados ocupa mais de dois terabytes (um terabyte é 10^{12} bytes de armazenamento) e está armazenado em 200 computadores diferentes (denominados servidores). Cerca de 15 milhões de visitantes acessam a Amazon.com todos os dias e utilizam o banco de dados para fazer compras. O banco de dados é continuamente atualizado à medida que novos livros e outros itens são acrescentados ao estoque e as quantidades em estoque são atualizadas à medida que as compras são feitas. Cerca de cem pessoas são responsáveis por manter o banco de dados da Amazon atualizado.

Um banco de dados pode ser gerado e mantido manualmente, ou pode ser computadorizado. Por exemplo, um catálogo de cartão de biblioteca é um banco de dados que pode ser criado e mantido manualmente. Um banco de dados computadorizado pode ser criado e mantido por um grupo de programas de aplicação escritos especificamente para essa tarefa ou por um sistema gerenciador de banco de dados. Vamos tratar apenas dos bancos de dados computadorizados neste livro.

Um **sistema gerenciador de banco de dados** (**SGBD** — Database Management System) é uma coleção de programas que permite aos usuários criar e manter um banco de dados. O SGBD é um *sistema de software de uso geral* que facilita o processo de

[1]O livro original utiliza a palavra *data* em singular e plural, pois isso é comum na literatura de banco de dados; o contexto determinará se ela está no singular ou no plural. (Em inglês padrão, *data* é usado para o plural e *datum,* para o singular.)

definição, *construção*, *manipulação* e *compartilhamento* de bancos de dados entre diversos usuários e aplicações. **Definir** um banco de dados envolve especificar os tipos, estruturas e restrições dos dados a serem armazenados. A definição ou informação descritiva do banco de dados também é armazenada pelo SGBD na forma de um catálogo ou dicionário, chamado de **metadados**. A **construção** do banco de dados é o processo de armazenar os dados em algum meio controlado pelo SGBD. A **manipulação** de um banco de dados inclui funções como consulta ao banco de dados para recuperar dados específicos, atualização do banco de dados para refletir mudanças no minimundo e geração de relatórios com base nos dados. O **compartilhamento** de um banco de dados permite que diversos usuários e programas acessem-no simultaneamente.

Um **programa de aplicação** acessa o banco de dados ao enviar consultas ou solicitações de dados ao SGBD. Uma **consulta**[2] normalmente resulta na recuperação de alguns dados; uma **transação** pode fazer que alguns dados sejam lidos e outros, gravados no banco de dados.

Outras funções importantes fornecidas pelo SGBD incluem *proteção* do banco de dados e sua *manutenção* por um longo período. A **proteção** inclui *proteção do sistema* contra defeitos (ou falhas) de hardware ou software e *proteção de segurança* contra acesso não autorizado ou malicioso. Um banco de dados grande pode ter um ciclo de vida de muitos anos, de modo que o SGBD precisa ser capaz de **manter** o sistema, permitindo que ele evolua à medida que os requisitos mudam com o tempo.

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

Para completar nossas definições iniciais, chamaremos a união do banco de dados com o software de SGBD de **sistema de banco de dados**. A Figura 1.1 ilustra alguns dos conceitos que discutimos até aqui.

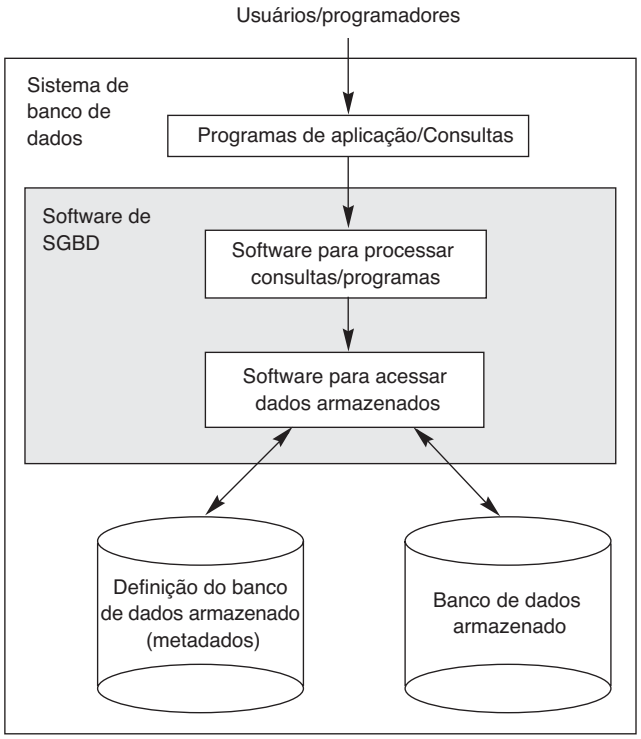

Figura 1.1
Diagrama simplificado de um ambiente de sistema de banco de dados.

1.2 Um exemplo

Vamos considerar um exemplo simples ao qual a maioria dos leitores pode estar acostumada: um banco de dados UNIVERSIDADE para manter informações referentes a alunos, disciplinas e notas em um ambiente universitário. A Figura 1.2 mostra a estrutura e alguns exemplos de dados para o banco de dados UNIVERSIDADE. O banco de dados está organizado como cinco arquivos, e cada um armazena **registros de dados** do mesmo tipo.[3] O arquivo ALUNO armazena dados sobre cada aluno, o arquivo disciplina armazena dados sobre cada disciplina, o arquivo TURMA armazena dados sobre cada turma de uma disciplina, o arquivo HISTORICO_ESCOLAR armazena as notas que os alunos recebem nas várias turmas que eles concluíram, e o arquivo PRE_REQUISITO armazena os pré-requisitos de cada disciplina.

Para *definir* esse banco de dados, precisamos especificar a estrutura dos registros de cada arquivo, determinando os diferentes tipos de **elementos de dados** a serem armazenados em cada registro. Na Figura 1.2, cada registro de ALUNO contém os dados que represen-

[2] O termo *consulta* (ou *query*), que originalmente significa uma pergunta ou uma pesquisa, é usado livremente para todos os tipos de interações com bancos de dados, incluindo a modificação dos dados.

[3] Usamos o termo *arquivo* informalmente aqui. Em um nível conceitual, um *arquivo* é uma *coleção* de registros que podem ou não estar ordenados.

ALUNO

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

DISCIPLINA

Nome_ disciplina	Numero_ disciplina	Creditos	Departamento
Introd. à ciência da computação	CC1310	4	CC
Estruturas de dados	CC3320	4	CC
Matemática discreta	MAT2410	3	MAT
Banco de dados	CC3380	3	CC

TURMA

Identificacao_ turma	Numero_ disciplina	Semestre	Ano	Professor
85	MAT2410	Segundo	07	Kleber
92	CC1310	Segundo	07	Anderson
102	CC3320	Primeiro	08	Carlos
112	MAT2410	Segundo	08	Chang
119	CC1310	Segundo	08	Anderson
135	CC3380	Segundo	08	Santos

HISTORICO_ESCOLAR

Numero_aluno	Identificacao_turma	Nota
17	112	B
17	119	C
8	85	A
8	92	A
8	102	B
8	135	A

PRE_REQUISITO

Numero_disciplina	Numero_pre_requisito
CC3380	CC3320
CC3380	MAT2410
CC3320	CC1310

Figura 1.2
Exemplo de banco de dados que armazena informações de aluno e disciplina.

tam o Nome, Numero_aluno, Tipo_aluno (como novato igual a '1', segundo ano igual a '2', e assim por diante) e Curso (como matemática igual a 'MAT' e ciência da computação igual a 'CC'); cada registro de DISCIPLINA contém os dados que representam o Nome_disciplina, Numero_disciplina, Creditos e Departamento (o departamento que oferece a disciplina); e assim por diante. Também precisamos especificar um **tipo de dado** para cada elemento de dado em um registro. Por exemplo, podemos especificar que o Nome de ALUNO é uma sequência de caracteres alfabéticos; Numero_aluno de ALUNO é um inteiro, e Nota de HISTORICO_ESCOLAR é um único caractere do conjunto {'A', 'B', 'C', 'D', 'F'}. Também podemos usar um esquema de codificação para representar os valores de um item de dados. Por exemplo, na Figura 1.2, representamos Tipo_aluno de um ALUNO como 1 para novato, 2 para segundo ano, 3 para júnior, 4 para sênior e 5 para aluno formado.

Para *construir* o banco de dados UNIVERSIDADE, armazenamos dados para representar cada aluno, disciplina, turma, histórico escolar e pré-requisito como um registro no arquivo apropriado. Observe que os registros nos diversos arquivos podem estar relacionados. Por exemplo, o registro para Silva no arquivo ALUNO está relacionado a dois registros no arquivo HISTORICO_ESCOLAR, que especifica as notas de Silva em duas turmas. De modo semelhante, cada registro no arquivo PRE_REQUISITO relaciona-se a dois registros de disciplina; um representando a disciplina e o outro representando o pré-requisito. A maioria dos bancos de dados de tamanho médio e grande inclui muitos tipos de registros e possui *muitos relacionamentos* entre os registros.

A *manipulação* do banco de dados envolve consulta e atualização. Alguns exemplos de consultas são os seguintes:

- Recuperar uma lista de todas as disciplinas e notas de 'Silva'.
- Listar os nomes dos alunos que realizaram a disciplina 'Banco de dados' oferecida no segundo semestre de 2008 e suas notas nessa turma.
- Listar os pré-requisitos do curso de 'Banco de dados'.

Alguns exemplos de atualizações incluem:

- Alterar o tipo de aluno de 'Silva' para segundo ano.
- Criar outra turma para a disciplina 'Banco de dados' para este semestre.
- Inserir uma nota 'A' para 'Silva' na turma 'Banco de dados' do último semestre.

Essas consultas e atualizações informais precisam ser especificadas corretamente na linguagem de consulta do SGBD antes de serem processadas.

Nesse estágio, é útil descrever o banco de dados como parte de uma tarefa maior conhecida como sistema de informação dentro de qualquer organização.

O departamento de Tecnologia da Informação (TI) de uma empresa projeta e mantém um sistema de informações que consiste em vários computadores, sistemas de armazenamento, software de aplicação e bancos de dados. O projeto de uma nova aplicação para um banco de dados existente ou de um novo banco de dados começa com uma fase chamada **especificação e análise de requisitos**. Esses requisitos são documentados com detalhes e transformados em um **projeto conceitual**, que pode ser representado e manipulado usando algumas ferramentas computadorizadas para que possa ser facilmente mantido, modificado e transformado em uma implementação de banco de dados. (Apresentaremos um modelo denominado Entidade-Relacionamento, no Capítulo 7, que é usado para essa finalidade.) O projeto é então traduzido para um **projeto lógico**, que pode ser expresso em um modelo de dados implementado em um SGBD comercial. (Neste livro, vamos enfatizar um modelo de dados conhecido como Modelo de Dados Relacional a partir do Capítulo 3. Essa é atualmente a técnica mais popular para projetar e implementar bancos de dados usando SGBDs relacionais.) O estágio final é o **projeto físico**, durante o qual outras especificações são fornecidas para armazenar e acessar o banco de dados. O projeto de banco de dados é implementado, alimentado com dados reais e mantido continuamente para refletir o estado do minimundo.

1.3 Características da abordagem de banco de dados

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

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

- Natureza de autodescrição de um sistema de banco de dados.
- Isolamento entre programas e dados, e abstração de dados.
- Suporte de múltiplas visões dos dados.
- Compartilhamento de dados e processamento de transação multiusuário.

Descrevemos cada uma dessas características em uma seção distinta. Discutiremos características adicionais dos sistemas de banco de dados nas seções 1.6 a 1.8.

1.3.1 Natureza de autodescrição de um sistema de banco de dados

Uma característica fundamental da abordagem de banco de dados é que seu sistema contém não apenas o próprio banco de dados, mas também uma definição ou descrição completa de sua estrutura e restrições. Essa definição é armazenada no catálogo do SGBD, que possui informações como a estrutura de cada arquivo, o tipo e o formato de armazenamento de cada item de dados e diversas restrições sobre os dados. A informação armazenada no catálogo é chamada de **metadados**, e descreve a estrutura do banco de dados principal (Figura 1.1).

O catálogo é usado pelo software de SGBD e também pelos usuários do banco de dados que precisam de informações sobre a estrutura do banco de dados. Um pacote de software de SGBD de uso geral não é escrito para uma aplicação de banco de dados específica. Portanto, ele precisa consultar o catálogo para conhecer a estrutura dos arquivos em um banco de dados específico, como o tipo e o formato dos dados que ele acessará. O software de SGBD precisa trabalhar de forma satisfatória com *qualquer quantidade de aplicações de banco de dados* — por exemplo, um banco de dados de universidade, de banco ou de uma empresa —, desde que sua definição esteja armazenada no catálogo.

No processamento de arquivos tradicional, a definição de dados normalmente faz parte dos próprios programas de aplicação. Logo, esses programas são forçados a trabalhar com apenas *um banco de dados específico*, cuja estrutura é declarada nos programas de aplicação. Por exemplo, um programa de aplicação

escrito em C++ pode ter declarações de estrutura ou classe, e um programa em COBOL tem instruções da divisão de dados para definir seus arquivos. Enquanto o software de processamento de arquivos pode acessar apenas bancos de dados específicos, o software de SGBD pode acessar diversos bancos de dados extraindo e usando as definições do catálogo de banco de dados.

Para o exemplo mostrado na Figura 1.2, o catálogo do SGBD armazenará as definições de todos os arquivos mostrados. A Figura 1.3 mostra alguns exemplos de entradas em um catálogo de banco de dados. Essas definições são especificadas pelo projetista antes da criação do banco de dados real e armazenadas no catálogo. Sempre que é feita uma solicitação para acessar, digamos, o Nome de um registro de ALUNO, o software de SGBD consulta o catálogo para determinar a estrutura do arquivo ALUNO e a posição e o tamanho do item de dado Nome dentro do registro de ALUNO. Ao contrário, em uma aplicação típica de processamento de arquivo, a estrutura do arquivo e, no caso extremo, a localização exata do Nome dentro de um registro de ALUNO já estão codificadas dentro de cada programa que acessa esse item de dados.

RELACOES

Nome_relacao	Numero_de_colunas
ALUNO	4
DISCIPLINA	4
TURMA	5
HISTORICO_ESCOLAR	3
PRE_REQUISITO	2

COLUNAS

Nome_coluna	Tipo_de_dado	Pertence_a_relacao
Nome	Caractere (30)	ALUNO
Numero_aluno	Caractere (4)	ALUNO
Tipo_aluno	Inteiro (1)	ALUNO
Curso	Tipo_curso	ALUNO
Nome_disciplina	Caractere (10)	DISCIPLINA
Numero_disciplina	XXXXNNNN	DISCIPLINA
....	....	....
....	....	....
....	....	....
Numero_pre_requisito	XXXXNNNN	PRE-REQUISITO

Figura 1.3
Exemplo de um catálogo para o banco de dados na Figura 1.2.

Nota: Tipo_curso é definido como um tipo enumerado com todas as matérias conhecidas. XXXXNNNN é usado para definir um tipo com quatro características alfanuméricas seguidas por quatro dígitos.

1.3.2 Isolamento entre programas e dados, e abstração de dados

No processamento de arquivos tradicional, a estrutura dos arquivos de dados está embutida nos programas de aplicação, de modo que quaisquer mudanças em sua estrutura podem exigir *alteração em todos os programas* que acessam esse arquivo. Ao contrário, os programas que acessam o SGBD não exigem tais mudanças na maioria dos casos. A estrutura dos arquivos de dados é armazenada no catálogo do SGBD separadamente dos programas de acesso. Chamamos essa propriedade de **independência de dados do programa.**

Por exemplo, um programa de acesso a arquivo pode ser escrito para acessar apenas registros de ALUNO da estrutura mostrada na Figura 1.4. Se quisermos acrescentar outro dado a cada registro de ALUNO, digamos Data_nascimento, esse programa não funcionará mais, e precisará ser alterado. Ao contrário, em um ambiente de SGBD, só precisamos mudar a descrição dos registros de ALUNO no catálogo (Figura 1.3) para refletir a inclusão do novo item de dado Data_nascimento; nenhum programa é alterado. Da próxima vez que o programa de SGBD consultar o catálogo, a nova estrutura dos registros de ALUNO será acessada e usada.

Em alguns tipos de sistemas de banco de dados, como os orientados a objeto e objeto-relacional (ver Capítulo 11), os usuários podem definir operações sobre dados como parte das definições do banco de dados. Uma **operação** (também chamada de *função* ou *método*) é especificada em duas partes. A *interface* (ou *assinatura*) de uma operação inclui o nome da operação e os tipos de dados de seus argumentos (ou parâmetros). A *implementação* (ou *método*) da operação é especificada separadamente e pode ser alterada sem afetar a interface. Os programas de aplicação do usuário podem operar sobre os dados invocando essas operações por meio de seus nomes e argumentos, independentemente de como as operações são implementadas. Isso pode ser chamado de **independência da operação do programa.**

Nome do item de dados	Posicionamento inicial no registro	Tamanho em caracteres (bytes)
Nome	1	30
Numero_aluno	31	4
Tipo_aluno	35	1
Curso	36	4

Figura 1.4
Formato de armazenamento interno para um registro de ALUNO, baseado no catálogo do banco de dados da Figura 1.3.

A característica que permite a independência de dados do programa e a independência da operação do programa é chamada de **abstração de dados**. Um SGBD oferece aos usuários uma **representação conceitual** de dados que não inclui muitos dos detalhes de como os dados são armazenados ou como as operações são implementadas. De maneira informal, um **modelo de dados** é um tipo de abstração de dados usado para oferecer essa representação conceitual. O modelo de dados usa conceitos lógicos, como objetos, suas propriedades e seus inter-relacionamentos, que podem ser mais fáceis para os usuários entenderem do que os conceitos de armazenamento de computador. Logo, o modelo de dados *oculta* os detalhes de armazenamento e implementação que não são do interesse da maioria dos usuários de banco de dados.

Por exemplo, reconsidere as figuras 1.2 e 1.3. A implementação interna de um arquivo pode ser definida por seu tamanho de registro — o número de caracteres (bytes) em cada registro — e cada item de dados pode ser especificado pelo byte inicial dentro de um registro e seu tamanho em bytes. O registro de ALUNO, assim, seria representado como mostra a Figura 1.4. Mas um típico usuário de banco de dados não está preocupado com a localização de cada item de dados dentro de um registro ou em seu tamanho; em vez disso, o usuário se preocupa se o valor é retornado corretamente, quando for feita uma referência ao Nome do ALUNO. Uma representação conceitual dos registros de ALUNO aparece na Figura 1.2. Muitos outros detalhes da organização do armazenamento do arquivo — como os caminhos de acesso especificados em um arquivo — podem ser ocultados dos usuários do banco de dados pelo SGBD. Discutiremos detalhes de armazenamento nos capítulos 17 e 18.

Na abordagem de banco de dados, a estrutura detalhada e a organização de cada arquivo são armazenadas no catálogo. Os usuários do banco de dados e os programas de aplicação se referem à representação conceitual dos arquivos, e o SGBD extrai os detalhes do armazenamento do arquivo do catálogo quando estes são necessários para os módulos de acesso a arquivo do SGBD. Muitos modelos de dados podem ser usados para oferecer essa abstração aos usuários do banco de dados. A primeira parte deste livro é dedicada à apresentação dos vários modelos de dados e dos conceitos que eles utilizam para abstrair a representação dos dados.

Nos bancos de dados orientados a objeto e objeto-relacional, o processo de abstração inclui não apenas a estrutura dos dados, mas também as operações sobre os dados. Essas operações oferecem uma abstração das atividades do minimundo comumente entendidas pelos usuários. Por exemplo, uma operação CALCULA_MEDIA pode ser aplicada ao objeto ALUNO para calcular a média das notas. Essas operações podem ser solicitadas pelas consultas do usuário ou por programas de aplicação sem ter que saber os detalhes de como as operações são implementadas. Nesse sentido, uma abstração da atividade do minimundo se torna disponível ao usuário como uma **operação abstrata**.

1.3.3 Suporte para múltiplas visões dos dados

Um banco de dados em geral tem muitos usuários, cada um podendo exigir um ponto de vista ou **visão** diferente do banco de dados. Uma visão (ou *view*) pode ser um subconjunto do banco de dados ou conter **dado virtual** que é derivado dos arquivos do banco de dados, mas não estão armazenados explicitamente. Alguns usuários não precisam saber se os dados a que se referem estão armazenados ou se são derivados. Um SGBD multiusuário, cujos usuários têm uma série de aplicações distintas, precisa oferecer facilidades para definir múltiplas visões. Por exemplo, um usuário do banco de dados da Figura 1.2 pode estar interessado apenas em acessar e imprimir o histórico escolar de cada aluno; a visão para esse usuário é mostrada na Figura 1.5(a). Um segundo usuário, que está interessado apenas em verificar se os alunos possuem todos os pré-requisitos de cada disciplina para a qual se inscreveram, pode requerer a visão apresentada na Figura 1.5(b).

1.3.4 Compartilhamento de dados e processamento de transação multiusuário

Um SGBD multiusuário, como o nome sugere, precisa permitir que múltiplos usuários acessem o banco de dados ao mesmo tempo. Isso é essencial se o dado para múltiplas aplicações está sendo integrado e mantido em um único banco de dados. O SGBD precisa incluir um software de **controle de concorrência** para garantir que vários usuários tentando atualizar o mesmo dado faça isso de uma maneira controlada, de modo que o resultado dessas atualizações seja correto. Por exemplo, quando vários agentes de viagem tentam reservar um assento em um voo de uma companhia aérea, o SGBD precisa garantir que cada assento só possa ser acessado por um agente de cada vez para que seja atribuído a um único passageiro. Esses tipos de aplicações geralmente são chamados de aplicações de **processamento de transação on-line** (**OLPT** — On-Line Transaction Processing). Um papel fundamental do software SGBD multiusuário é garantir que as transações concorrentes operem de maneira correta e eficiente.

(a)

DADO_ESCOLAR

Nome_aluno	Historico_escolar_aluno				
	Numero_disciplina	Nota	Semestre	Ano	Identificacao_turma
Silvah	CC1310	C	Segundo	08	119
	MAT2410	B	Segundo	08	112
Braga	MAT2410	A	Segundo	07	85
	CC1310	A	Segundo	07	92
	CC3320	B	Primeiro	08	102
	CC3380	A	Segundo	08	135

(b)

PRE_REQUISITO_DISCIPLINA

Nome_disciplina	Numero_disciplina	Pre_requisitos
Banco de dados	CC3380	CC3320
		MAT2410
Estrutura de dados	CC3320	CC1310

Figura 1.5
Duas visões derivadas do banco de dados da Figura 1.2. (a) A visão do HISTORICO_ESCOLAR. (b) A visão do PRE_REQUISITO_DISCIPLINA.

O conceito de **transação** tem se tornado fundamental para muitas aplicações de banco de dados. Uma transação é um *programa em execução* ou *processo* que inclui um ou mais acessos ao banco de dados, como a leitura ou atualização de seus registros. Uma transação executa um acesso logicamente correto a um banco de dados quando ela é executada de forma completa e sem interferência de outras transações. O SGBD precisa impor várias propriedades da transação. A propriedade de **isolamento** garante que cada transação pareça executar isoladamente das demais, embora centenas de transações possam estar executando concorrentemente. A propriedade de **atomicidade** garante que todas as operações em uma transação sejam executadas ou que nenhuma seja. Discutiremos sobre transações em detalhes na Parte 9.

As características anteriores são importantes para distinguir um SGBD de um software tradicional de processamento de arquivo. Na Seção 1.6, discutiremos recursos adicionais que caracterizam um SGBD. Primeiro, porém, vamos categorizar os diferentes tipos de pessoas que trabalham em um ambiente de sistema de banco de dados.

1.4 Atores em cena

Para um pequeno banco de dados pessoal, como a lista de endereços discutida na Seção 1.1, uma pessoa normalmente define, constrói e manipula o banco de dados, sem compartilhamento. Porém, em grandes organizações, muitas pessoas estão envolvidas no projeto, no uso e na manutenção de um grande banco de dados, com centenas de usuários. Nesta seção, identificamos as pessoas cujas funções envolvem o uso diário de um grande banco de dados; nós os chamamos de *atores em cena*. Na Seção 1.5, consideraremos as pessoas que podem ser chamadas de *trabalhadores dos bastidores* — aqueles que trabalham para manter o ambiente do sistema de banco de dados, mas que não estão ativamente interessados em seu conteúdo como parte de sua função diária.

1.4.1 Administradores de banco de dados

Em qualquer organização onde muitas pessoas utilizam os mesmos recursos, há uma necessidade de um administrador principal para supervisionar e gerenciar tais recursos. Em um ambiente de banco de dados, o recurso principal é o próprio banco de dados, e o recurso secundário é o SGBD e os softwares relacionados. A administração desses recursos é de responsabilidade do **administrador de banco de dados** (**DBA —** *database administrator*). O DBA é responsável por autorizar o acesso ao banco de dados, coordenar e monitorar seu uso e adquirir recursos de software e hardware conforme a necessidade. Também é responsável por problemas como falhas na segurança e demora no tempo de resposta do sistema. Em grandes organizações, ele é auxiliado por uma equipe que executa essas funções.

1.4.2 Projetistas de banco de dados

Os **projetistas de banco de dados** são responsáveis por identificar os dados a serem armazenados e escolher estruturas apropriadas para representar e armazenar esses dados. Essas tarefas são realizadas principalmente antes que o banco de dados esteja realmente implementado e populado com dados. É responsabilidade dos projetistas de banco de dados se comunicar com todos os potenciais usuários a fim de entender suas necessidades e criar um projeto que as atenda. Em muitos casos, os projetistas estão na equipe de DBAs e podem receber outras responsabilidades após o projeto do banco de dados estar concluído. Os projetistas de banco de dados normalmente interagem com cada potencial grupo de usuários e desenvolvem **visões** do banco de dados que cumprem os requisitos de dados e processamento desses grupos. Cada visão é então analisada e *integrada* às visões de outros grupos de usuários.

O projeto final do banco de dados precisa ser capaz de atender às necessidades de todos os grupos de usuários.

1.4.3 Usuários finais

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

- **Usuários finais casuais** ocasionalmente acessam o banco de dados, mas podem precisar de diferentes informações a cada vez. Utilizam uma linguagem sofisticada de consulta ao banco de dados para especificar suas necessidades e normalmente são gerentes de nível intermediário ou alto, ou outros usuários ocasionais.
- **Usuários finais iniciantes** ou **paramétricos** compõem uma grande parte dos usuários finais do banco de dados. Sua função principal gira em torno de consultar e atualizar o banco de dados constantemente, usando tipos padrão de consultas e atualizações — denominadas **transações programadas** — que foram cuidadosamente programadas e testadas. As tarefas que esses usuários realizam são variadas:
  - Caixas de banco verificam saldos de conta e realizam saques, depósitos, pagamentos etc.
  - Agentes de companhias aéreas, hotéis e locadoras de automóveis verificam a disponibilidade de determinada solicitação e fazem reservas.
  - Funcionários nas estações de recebimento de transportadoras inserem identificações de pacotes por códigos de barra e informações descritivas por meio de etiquetas para atualizar um banco de dados central de pacotes recebidos e em trânsito.
- **Usuários finais sofisticados** incluem engenheiros, cientistas, analistas de negócios e outros que estão profundamente familiarizados com as facilidades do SGBD a ponto de implementar as próprias aplicações para que atendam a suas necessidades complexas.
- **Usuários isolados** mantêm bancos de dados pessoais usando pacotes de programas prontos, que oferecem interfaces de fácil utilização, baseadas em menus ou gráficos. Um exemplo é o usuário de um pacote de cálculos de impostos, que armazena uma série de dados financeiros pessoais para fins de declaração de imposto.

Um SGBD típico oferece múltiplas facilidades para acessar um banco de dados. Usuários iniciantes precisam aprender muito pouco sobre as facilidades oferecidas pelo SGBD; eles simplesmente têm de entender as interfaces de usuário das transações padrão projetadas e implementadas para seu uso. Os usuários casuais aprendem apenas algumas facilidades que podem usar repetidamente. Usuários sofisticados tentam aprender a maioria das facilidades do SGBD para satisfazer suas necessidades complexas. Usuários isolados costumam se tornar especialistas no uso de um pacote de software específico.

1.4.4 Analistas de sistemas e programadores de aplicações (engenheiros de software)

Analistas de sistemas identificam as necessidades dos usuários finais, especialmente os iniciantes e paramétricos, e definem as especificações das transações padrão que atendam a elas. Os **programadores de aplicações** implementam essas especificações como programas; depois, eles testam, depuram, documentam e mantêm essas transações programadas. Esses analistas e programadores — também conhecidos como **engenheiros de software** e **desenvolvedores de sistemas de software** — devem estar familiarizados com todo o conjunto de capacidades fornecido pelo SGBD para realizarem suas tarefas.

1.5 Trabalhadores dos bastidores

Além daqueles que projetam, usam e administram um banco de dados, há outros associados ao projeto, desenvolvimento e operação do *software e ambiente de sistema* do SGBD. Essas pessoas normalmente não estão interessadas no conteúdo do banco de dados em si. Vamos chamá-las de *trabalhadores dos bastidores*, e elas estão incluídas nas seguintes categorias:

- **Projetistas e implementadores de sistema de SGBD** projetam e implementam os módulos e as interfaces do SGBD como um pacote de software. Um SGBD é um sistema muito complexo, que consiste em muitos componentes, ou **módulos**, incluindo módulos para implementação do catálogo, processamento de linguagem de consulta, processamento de interface, acesso e buffering de dados, controle de concorrência e tratamento de recuperação e segurança de dados. O SGBD precisa realizar a interface com outros sistemas de software, como o sistema operacional, e compiladores para diversas linguagens de programação.

- **Desenvolvedores de ferramentas** projetam e implantam **ferramentas** — os pacotes de software que facilitam a modelagem e o projeto do banco de dados, o projeto do sistema de banco de dados e a melhoria no desempenho. Ferramentas são pacotes opcionais que, em geral, são adquiridos separadamente. Elas incluem pacotes para projeto de banco de dados, monitoramento de desempenho, linguagem natural ou interfaces gráficas, protótipo, simulação e geração de dados de teste. Em muitos casos, fornecedores de software independentes desenvolvem e comercializam essas ferramentas.
- **Operadores e pessoal de manutenção** (pessoal de administração de sistemas) são responsáveis pela execução e manutenção do ambiente de hardware e software para o sistema de banco de dados.

Embora essas categorias de trabalhadores dos bastidores sejam instrumento para tornar o sistema de banco de dados disponível aos usuários finais, eles não costumam utilizar o conteúdo do banco de dados para fins pessoais.

1.6 Vantagens de usar a abordagem de SGBD

Nesta seção, discutiremos algumas das vantagens de usar um bom SGBD e as capacidades que ele deve possuir. Essas capacidades estão além das quatro principais características discutidas na Seção 1.3. O DBA deve utilizá-las para cumprir uma série de objetivos relacionados ao projeto, à administração e ao uso de um grande banco de dados multiusuário.

1.6.1 Controlando a redundância

No desenvolvimento de software tradicional, utilizando processamento de arquivo, cada grupo de usuários mantém os próprios arquivos para tratamento de suas aplicações de processamento de dados. Por exemplo, considere o exemplo do banco de dados UNIVERSIDADE da Seção 1.2; aqui, dois grupos de usuários podem ser o pessoal do departamento de registro acadêmico e departamento de finanças. Na técnica tradicional, cada grupo mantém de maneira independente os arquivos sobre os alunos. O departamento financeiro mantém dados sobre o registro e informações relacionadas a faturas, enquanto o departamento de registro acadêmico acompanha as disciplinas e as notas dos alunos. Outros grupos podem duplicar ainda mais alguns ou todos os dados nos próprios arquivos.

Essa **redundância** causada ao armazenar os mesmos dados várias vezes gera diversos problemas. Primeiro, é preciso realizar uma única atualização lógica — como a entrada de dados sobre um novo aluno — várias vezes: uma para cada arquivo onde o dado do aluno é registrado. Isso ocasiona uma *duplicação de esforço*. Segundo, o *espaço de armazenamento é desperdiçado* quando o mesmo dado é armazenado repetidamente, e esse problema pode ser sério para grandes bancos de dados. Terceiro, os arquivos que representam os mesmos dados podem tornar-se *inconsistentes*. Isso porque uma atualização é aplicada a alguns dos arquivos, mas não a outros. Mesmo que uma atualização — como a inclusão de um novo aluno — seja aplicada a todos os arquivos apropriados, os dados referentes ao aluno ainda podem ser *inconsistentes* porque as atualizações são aplicadas de maneira independente pelos grupos de usuários. Por exemplo, um grupo de usuários pode entrar com a data de nascimento de um aluno incorretamente como '19/01/1988', enquanto outros grupos de usuários podem inserir o valor correto '29/01/1988'.

Na abordagem de banco de dados, as visões de diferentes grupos de usuários são integradas durante o projeto. O ideal é que tenhamos um projeto que armazena cada item de dados lógico — como o nome ou a data de nascimento de um aluno — em *apenas um lugar* no banco de dados. Isso é conhecido como **normalização de dados**, e garante consistência e economia de espaço de armazenamento (a normalização de dados é descrita na Parte 6 do livro). Porém, na prática, às vezes é necessário usar a **redundância controlada** para melhorar o desempenho das consultas. Por exemplo, podemos armazenar Nome_aluno e Numero_disciplina redundantemente em um arquivo HISTORICO_ESCOLAR [Figura 1.6(a)] porque, sempre que recuperamos um registro de HISTORICO_ESCOLAR, queremos recuperar o nome do aluno e o número da disciplina juntamente com a nota, o número do aluno e o identificador de turma. Colocando todos os dados juntos, não precisamos pesquisar vários arquivos para coletar esses dados. Isso é conhecido como **desnormalização**. Nesses casos, o SGBD deve ter a capacidade de *controlar* essa redundância a fim de proibir inconsistências entre os arquivos. Isso pode ser feito verificando automaticamente se os valores de Nome_aluno-Numero_aluno em qualquer registro de HISTORICO_ESCOLAR na Figura 1.6(a) combinam com um dos valores de Nome-Numero_aluno de um registro de ALUNO (Figura 1.2). De modo semelhante, os valores de Identificacao_turma-Numero_disciplina de HISTORICO_ESCOLAR podem ser verificados em registros de TURMA. Essas verificações podem ser especificadas no SGBD durante o projeto do banco de dados e impostas automaticamente pelo SGBD sempre que o arquivo HISTORICO_ESCOLAR for atualizado. A Figura 1.6(b) mostra um registro de HISTORICO_ESCOLAR incoerente com o arquivo ALUNO na Figura 1.2; esse tipo de erro pode ser inserido se a redundância *não for controlada*. Você consegue identificar a parte inconsistente?

(a)

HISTORICO_ESCOLAR

Numero_aluno	Nome_aluno	Identificacao_turma	Numero_disciplina	Nota
17	Silva	112	MAT2410	B
17	Silva	119	CC1310	C
8	Braga	85	MAT2410	A
8	Braga	92	CC1310	A
8	Braga	102	CC3320	B
8	Braga	135	CC3380	A

(b)

HISTORICO_ESCOLAR

Numero_aluno	Nome_aluno	Identificacao_turma	Numero_disciplina	Nota
17	Braga	112	MAT2410	B

Figura 1.6
Armazenamento redundante de Nome_aluno e Nome_disciplina em HISTORICO_ESCOLAR. (a) Dados consistentes. (b) Registro inconsistente.

1.6.2 Restringindo o acesso não autorizado

Quando vários usuários compartilham um grande banco de dados, é provável que a maioria deles não esteja autorizada a acessar todas as informações nele contidas. Por exemplo, dados financeiros normalmente são considerados confidenciais, e somente pessoas autorizadas têm permissão para acessá-los. Além disso, alguns usuários só podem ter permissão para recuperar dados, enquanto outros podem recuperar e atualizar. Logo, o tipo de operação de acesso — recuperação ou atualização — também deve ser controlado. Em geral, os usuários ou grupos de usuários recebem números de conta protegidos por senhas, que podem usar para acessar o banco de dados. Um SGBD deve oferecer um **subsistema de segurança e autorização**, que o DBA utiliza para criar contas e especificar suas restrições. Então, o SGBD deve impor essas restrições automaticamente. Observe que podemos aplicar controles semelhantes ao software de SGBD. Por exemplo, somente o DBA está autorizado a usar certo **software privilegiado**, como o software para criar contas. De modo semelhante, usuários paramétricos podem ter permissão para acessar o banco de dados apenas por meio de transações programadas predefinidas, desenvolvidas para seu uso.

1.6.3 Oferecendo armazenamento persistente para objetos do programa

Os bancos de dados podem ser usados para oferecer **armazenamento persistente** para objetos e estruturas de dados do programa. Esse é um dos principais motivos para a existência de **sistemas de banco de dados orientados a objeto**. Linguagens de programação normalmente possuem estruturas de dados complexas, como tipos de registro em Pascal ou definições de classe em C++ ou Java. Os valores das variáveis de programa ou estruturas dos objetos são descartados quando um programa termina, a menos que o programador os armazene explicitamente em arquivos permanentes, o que, em geral, envolve converter essas estruturas complexas em um formato adequado para armazenamento de arquivo. Quando surge a necessidade de ler esses dados mais uma vez, o programador precisa converter do formato de arquivo para a variável de programa ou estrutura de objeto. Os sistemas de banco de dados orientados a objeto são compatíveis com linguagens de programação, como C++ e Java, e o software de SGBD realiza automaticamente quaisquer conversões necessárias. Assim, um objeto complexo em C++ pode ser armazenado de forma permanente em um SGBD orientado a objeto. Esse objeto é considerado **persistente**, pois sobrevive ao término da execução e pode ser recuperado mais tarde diretamente por outro programa C++.

O armazenamento persistente de objetos de programa e estruturas de dados é uma função importante dos sistemas de banco de dados. Os sistemas tradicionais sofrem com frequência do chamado **problema de divergência de impedância**, pois as estruturas de dados fornecidas pelo SGBD são incompatíveis com as estruturas de dados da linguagem de programação. Os sistemas de banco de dados orientados a objeto em geral oferecem **compatibilidade** da estrutura de dados com uma ou mais linguagens de programação orientadas a objeto.

1.6.4 Oferecendo estruturas de armazenamento e técnicas de pesquisa para o processamento eficiente de consulta

Os sistemas de banco de dados precisam oferecer capacidades para *executar consultas e atualizações de modo eficiente*. Como o banco de dados costuma ser armazenado em disco, o SGBD precisa oferecer estruturas de dados e técnicas de pesquisa especializadas para agilizar a busca dos registros desejados no disco. Arquivos auxiliares, denominados **índices**, são usados para essa finalidade. Os índices normalmente são baseados em estruturas de dados em árvore ou estrutura de dados em *hash*, que são modificadas de maneira adequada para a pesquisa no disco. Para processar os registros de banco de dados necessários por uma consulta em particular, eles precisam ser copiados do disco para a memória principal. Portanto,

o SGBD frequentemente tem um módulo de **buffering** ou **caching** que mantém partes do banco de dados nos buffers de memória principais. Em geral, o sistema operacional é responsável pelo buffering do disco para a memória. Contudo, como o buffering de dados é essencial para o desempenho do SGBD, a maioria desses sistemas realiza o próprio buffering de dados.

O módulo de **processamento e otimização de consulta** do SGBD é responsável por escolher um plano de execução eficiente para cada consulta, com base nas estruturas de armazenamento existentes. A escolha de quais índices criar e manter faz parte do *projeto e ajuste de banco de dados físico*, que é uma das responsabilidades da equipe de DBAs. Discutiremos sobre processamento de consulta, otimização e ajuste na Parte 8 do livro.

1.6.5 Oferecendo backup e recuperação

Um SGBD precisa oferecer recursos para recuperar-se de falhas de hardware ou software. Seu **subsistema de backup e recuperação** é responsável por isso. Por exemplo, se o sistema do computador falhar no meio de uma transação de atualização complexa, o subsistema de recuperação é responsável por garantir que o banco de dados seja restaurado ao estado em que estava antes da transação ser executada. Como alternativa, o subsistema de recuperação poderia garantir que a transação seja reiniciada no ponto em que foi interrompida, de modo que seu efeito completo seja registrado no banco de dados. O backup de disco também é necessário no caso de uma falha de disco catastrófica. Discutiremos a respeito do backup e recuperação no Capítulo 23.

1.6.6 Oferecendo múltiplas interfaces do usuário

Uma vez que muitos tipos de usuários, com diversos níveis de conhecimento técnico, utilizam um banco de dados, um SGBD deve oferecer uma variedade de interfaces de usuário. Essas incluem linguagens de consulta para usuários casuais, interfaces de linguagem de programação para programadores de aplicação, formulários e códigos de comando para usuários paramétricos e interfaces controladas por menu e de linguagem natural para usuários isolados. As interfaces no estilo de formulários e de menus normalmente são conhecidas como **interfaces gráficas do usuário** (**GUIs** — Graphical User Interfaces). Existem muitas linguagens e ambientes especializados para especificar GUIs. Recursos para oferecer interfaces GUI para um banco de dados na Web — ou habilitar um banco de dados para a Web — também são muito comuns.

1.6.7 Representando relacionamentos complexos entre dados

Um banco de dados pode incluir muitas variedades de dados que estão inter-relacionados de diversas maneiras. Considere o exemplo mostrado na Figura 1.2. O registro de 'Braga' no arquivo ALUNO está relacionado a quatro registros no arquivo HISTORICO_ESCOLAR. De modo semelhante, cada registro de turma está relacionado a um registro de disciplina e a uma série de registros de HISTORICO_ESCOLAR — um para cada aluno que concluiu a turma. Um SGBD precisa ter a capacidade de representar uma série de relacionamentos complexos entre os dados, definir novos relacionamentos à medida que eles surgem e recuperar e atualizar dados relacionados de modo fácil e eficaz.

1.6.8 Impondo restrições de integridade

A maioria das aplicações de banco de dados possui certas **restrições de integridade** que devem ser mantidas para os dados. Um SGBD deve oferecer capacidades para definir e impor tais restrições. O tipo mais simples de restrição de integridade envolve especificar um tipo de dado para cada item de dado. Por exemplo, na Figura 1.3, especificamos que o valor do item de dados Tipo_aluno em cada registro de ALUNO deve ser um inteiro de um dígito e que o valor de Nome precisa ser um alfanumérico de até 30 caracteres. Para restringir o valor de Tipo_aluno entre 1 e 5, seria preciso uma restrição adicional, que não aparece no catálogo atual. Um tipo de restrição mais complexo, que ocorre com frequência, envolve especificar que um registro em um arquivo deve estar relacionado a registros em outros arquivos. Por exemplo, na Figura 1.2, podemos especificar que *cada registro de turma deve estar relacionado a um registro de disciplina*. Isso é conhecido como restrição de **integridade referencial**. Outro tipo de restrição especifica a exclusividade sobre valores de item de dados, como *cada registro de disciplina deverá ter um valor exclusivo para Numero_disciplina*. Isso é conhecido como uma restrição de **chave** ou **singularidade**. Tais restrições são derivadas do significado ou da **semântica** dos dados e do minimundo que eles representam. É responsabilidade dos projetistas do banco de dados identificar restrições de integridade durante o projeto. Algumas restrições podem ser especificadas ao SGBD e impostas automaticamente. Outras podem ter que ser verificadas por programas de atualização ou no momento da entrada de dados. Em geral, para grandes aplicações, é comum chamar essas restrições de **regras de negócio**.

Um item de dados pode ser inserido erroneamente e ainda satisfazer as restrições de integridade especificadas. Por exemplo, se um aluno recebe uma nota 'A', mas uma nota 'C' é inserida no banco de dados, o SGBD *não pode* descobrir esse erro automaticamente,

pois 'C' é um valor válido para o tipo de dados Nota. Esses erros de entrada de dados só podem ser descobertos manualmente (quando o aluno recebe a nota e reclama) e corrigidos posteriormente, atualizando o banco de dados. Porém, uma nota 'Z' seria rejeitada automaticamente pelo SGBD, pois 'Z' não é um valor válido para o tipo de dado Nota. Quando discutirmos cada modelo de dados nos próximos capítulos, vamos apresentar regras que pertencem a esse modelo de maneira implícita. Por exemplo, no modelo Entidade-Relacionamento, no Capítulo 7, um relacionamento deve envolver pelo menos duas entidades. Essas regras são **regras inerentes** do modelo de dados e assumidas de maneira automática, para garantir a validade do modelo.

1.6.9 Permitindo dedução e ações usando regras

Alguns sistemas oferecem capacidades para definir *regras de dedução* (ou *inferência*) para *deduzir* novas informações com base nos fatos armazenados no banco de dados. Esses sistemas são chamados de **sistemas de banco de dados dedutivos**. Por exemplo, pode haver regras complexas na aplicação do minimundo para determinar quando um aluno está em época de prova. Estas podem ser especificadas *declarativamente* como **regras** que, quando compiladas e mantidas pelo SGBD, podem determinar todos os alunos em época de prova. Em um SGBD tradicional, um *código de programa de procedimento* explícito teria de ser escrito para dar suporte a tais aplicações. Mas, se as regras do minimundo mudarem, geralmente é mais conveniente mudar as regras de dedução declaradas do que recodificar programas de procedimento. Nos sistemas de banco de dados relacionais de hoje é possível associar **gatilhos** (ou **triggers**) a tabelas. Um gatilho é uma forma de regra ativada por atualizações na tabela, que resulta na realização de algumas operações adicionais em algumas outras tabelas, envio de mensagens, e assim por diante. Procedimentos mais elaborados para impor regras são popularmente chamados de **procedimentos armazenados** (ou **stored procedures**); eles se tornam parte da definição geral de banco de dados e são chamados de forma apropriada quando certas condições são atendidas. A funcionalidade mais poderosa é fornecida por **sistemas de banco de dados ativos**, que oferecem regras ativas que podem automaticamente iniciar ações quando ocorrem certos eventos e condições.

1.6.10 Implicações adicionais do uso da abordagem de banco de dados

Esta seção discute algumas implicações adicionais do uso da abordagem de banco de dados que pode beneficiar a maioria das organizações.

Potencial para garantir padrões. A técnica de banco de dados permite que o DBA defina e imponha o uso de padrões entre os usuários de banco de dados em uma grande organização. Isso facilita a comunicação e a cooperação entre seus vários departamentos, projetos e usuários dentro da organização. Podem ser definidos padrões para nomes e formatos dos elementos de dados, formatos de exibição, estruturas de relatório, terminologia, e assim por diante. O DBA pode impor padrões em um ambiente de banco de dados centralizado mais facilmente do que em um ambiente onde cada grupo de usuários tem controle sobre os próprios arquivos de dados e software.

Tempo reduzido para desenvolvimento de aplicação Um importante recurso de venda da abordagem de banco de dados é que o desenvolvimento de uma nova aplicação — como a recuperação de certos dados para impressão de um novo relatório — leva muito pouco tempo. Projetar e implementar um grande banco de dados multiusuário do zero pode levar mais tempo do que escrever uma única aplicação de arquivo especializada. Porém, quando um banco de dados está pronto e funcionando, geralmente é preciso muito menos tempo para criar outras aplicações usando as facilidades do SGBD. O tempo de desenvolvimento usando um SGBD é estimado como sendo um sexto a um quarto daquele para um sistema de arquivo tradicional.

Flexibilidade. Pode ser necessário mudar a estrutura de um banco de dados à medida que as necessidades mudam. Por exemplo, pode aparecer um novo grupo de usuários precisando de informações atualmente não incluídas no banco de dados. Em resposta, pode ser preciso acrescentar um arquivo ao banco de dados ou estender os elementos de dados em um arquivo existente. Os SGBDs modernos permitem certos tipos de mudanças evolucionárias na estrutura do banco de dados sem afetar os dados armazenados e os programas de aplicação existentes.

Disponibilidade de informações atualizadas. Um SGBD torna o banco de dados disponível a todos os usuários. Assim que a atualização de um usuário é aplicada ao banco de dados, todos os outros usuários podem vê-la imediatamente. Essa disponibilidade de informações atualizadas é essencial para muitas aplicações de processamento de transação, como sistemas de reserva ou bancos de dados bancários, e ela é possibilitada pelos subsistemas de controle de concorrência e recuperação de um SGBD.

Economias de escala. A técnica de SGBD permite a consolidação de dados e aplicações, reduzindo assim a quantidade de sobreposição desperdiçada entre as atividades do pessoal de processamento de dados em diferen-

tes projetos ou departamentos, bem como as redundâncias entre as aplicações. Isso permite que a organização inteira invista em processadores, dispositivos de armazenamento ou mecanismos de comunicação mais poderosos, em vez de cada departamento ter de comprar o próprio equipamento (menor desempenho), o que reduz os custos gerais de operação e gerenciamento.

1.7 Uma breve história das aplicações de banco de dados

Agora, vamos apresentar uma breve visão histórica das aplicações que usam SGBDs e como elas motivaram o desenvolvimento de novos tipos de sistemas de banco de dados.

1.7.1 Antigas aplicações de banco de dados usando sistemas hierárquicos e de rede

Muitas aplicações de banco de dados antigas mantinham os registros em grandes organizações, como corporações, universidades, hospitais e bancos. Em muitas delas, existia grande quantidade de registros com estrutura semelhante. Por exemplo, em uma aplicação de universidade, informações semelhantes seriam mantidas para cada aluno, cada disciplina, cada histórico escolar, e assim por diante. Também existiam muitos tipos de registros e muitos inter-relacionamentos entre eles.

Um dos principais problemas com os sistemas de banco de dados antigos era a mistura de relacionamentos conceituais com o armazenamento e posicionamento físico dos registros no disco. Logo, esses sistemas não ofereciam capacidades suficientes para *abstração de dados* e *independência entre dados e programas*. Por exemplo, as notas de determinado aluno poderiam ser armazenadas fisicamente próximas do registro do aluno. Embora isso fornecesse um acesso muito eficiente para as consultas e transações originais com as quais o banco de dados foi projetado para lidar, não oferecia flexibilidade suficiente para acessar registros de modo eficiente quando novas consultas e transações fossem identificadas. Em particular, novas consultas que exigiam uma organização de armazenamento diferente para o processamento eficiente eram muito difíceis de implementar de modo eficaz. Também era muito trabalhoso reorganizar o banco de dados quando eram feitas mudanças nos requisitos da aplicação.

Outra deficiência dos sistemas antigos era que eles ofereciam apenas interfaces da linguagem de programação. Isso tornava demorada e cara a implementação de novas consultas e transações, pois novos programas tinham de ser escritos, testados e depurados. A maioria desses sistemas de banco de dados era implantada em computadores mainframes grandes e caros, começando em meados da década de 1960 e continuando nos anos 1970 e 1980. Os principais tipos dos primeiros sistemas eram baseados em três paradigmas principais: sistemas hierárquicos, sistemas baseados em modelo de rede e sistemas de arquivo invertidos.

1.7.2 Oferecendo abstração de dados e flexibilidade de aplicação com bancos de dados relacionais

Os bancos de dados relacionais foram propostos originalmente para separar o armazenamento físico dos dados de sua representação conceitual e para fornecer uma base matemática para a representação e a consulta dos dados. O modelo de dados relacional também introduziu linguagens de consulta de alto nível, que ofereciam uma alternativa às interfaces de linguagem de programação, tornando muito mais rápida a escrita de novas consultas. A representação relacional dos dados é semelhante ao exemplo que apresentamos na Figura 1.2. Os sistemas relacionais visavam inicialmente atender às mesmas aplicações dos sistemas mais antigos, e forneciam flexibilidade para desenvolver rapidamente novas consultas e reorganizar o banco de dados à medida que os requisitos mudavam. Logo, a *abstração de dados* e a *independência entre dados e programas* eram mais desenvolvidas em comparação com os sistemas anteriores.

Os sistemas relacionais experimentais, desenvolvidos no final da década de 1970, e os sistemas de gerenciamento de bancos de dados relacionais (SGBDR), introduzidos na década de 1980, eram muito lentos, pois não usavam ponteiros de armazenamento físico ou posicionamento de registro para acessar registros de dados relacionados. Com o desenvolvimento de novas técnicas de armazenamento houve uma melhora no desempenho do processamento e otimização de consulta. Por fim, os bancos de dados relacionais se tornaram o tipo de sistema de banco de dados dominante para aplicações tradicionais. Eles agora existem em quase todos os tipos de computadores, desde os menores modelos pessoais até grandes servidores.

1.7.3 Aplicações orientadas a objeto e a necessidade de bancos de dados mais complexos

O surgimento de linguagens de programação orientadas a objeto no final da década de 1980 e a necessidade de armazenar e compartilhar objetos complexos e estruturados levou ao desenvolvimento

de Bancos de Dados Orientados a Objeto (BDOOs). Inicialmente, os BDOOs eram considerados um concorrente dos bancos de dados relacionais, pois forneciam estruturas de dados mais gerais. Eles também incorporavam muitos dos paradigmas úteis orientados a objeto, como tipos de dados abstratos, encapsulamento de operações, herança e identidade de objeto. Porém, a complexidade do modelo e a falta de um padrão inicial contribuíram para seu uso limitado. Eles agora são usados principalmente em aplicações especializadas, como projeto de engenharia, publicação de multimídia e sistemas de manufatura. Apesar das expectativas de que eles causarão um grande impacto, atualmente sua penetração geral no mercado de produtos de banco de dados permanece abaixo dos cinco por cento. Além disso, muitos conceitos orientados a objeto foram incorporados nas versões mais recentes dos SGBDs relacionados, levando a sistemas de gerenciamento de banco de dados objeto-relacional, conhecidos como SGBDORs.

1.7.4 Intercâmbio de dados na Web para comércio eletrônico usando XML

A World Wide Web oferece uma grande rede de computadores interconectados. Os usuários podem criar documentos usando uma linguagem de publicação na Web, como HyperText Markup Language (HTML ou, em português, linguagem de marcação de hipertexto), e armazenar esses documentos em servidores Web, onde outros usuários (clientes) podem acessá-los. Os documentos podem ser vinculados por meio de **hyperlinks**, que são indicadores para outros documentos. Na década de 1990, o comércio eletrônico (e-commerce) surgiu como uma importante aplicação da Web. Rapidamente ficou visível que partes da informação nas páginas Web de e-commerce eram, com frequência, dados extraídos dinamicamente de SGBDs. Diversas técnicas foram desenvolvidas para permitir o intercâmbio de dados na Web. Atualmente, a eXtended Markup Language (XML, em português, linguagem de marcadores extensível) é considerada o principal padrão para intercâmbio entre diversos tipos de bancos de dados e páginas Web. A XML combina conceitos dos modelos usados nos sistemas de documentos com os conceitos de modelagem de banco de dados. O Capítulo 12 é dedicado à discussão sobre a XML.

1.7.5 Estendendo as capacidades do banco de dados para novas aplicações

O sucesso dos sistemas de banco de dados nas aplicações tradicionais encorajou os desenvolvedores de outros tipos de aplicações a tentarem utilizá-los. Essas aplicações tradicionalmente usavam suas próprias estruturas especializadas de arquivo e dados. Os sistemas de banco de dados agora oferecem extensões para dar melhor suporte às necessidades especializadas para algumas dessas aplicações. A seguir estão alguns exemplos dessas aplicações:

- Aplicações **científicas** que armazenam grande quantidade de dados resultantes de experimentos científicos em áreas como a física de alta energia, o mapeamento do genoma humano e a descoberta de estruturas de proteínas.
- Armazenamento e recuperação de **imagens**, incluindo notícias escaneadas e fotografias pessoais, imagens fotográficas de satélite e imagens de procedimentos médicos, como raios X e IRMs (imagens por ressonância magnética).
- Armazenamento e recuperação de **vídeos**, como filmes, e **clipes de vídeo** de notícias ou de câmeras digitais pessoais.
- Aplicações de **mineração de dados** (ou **data mining**), que analisam grande quantidade de dados procurando as ocorrências de padrões ou relacionamentos específicos, e identificação de padrões incomuns em áreas como uso de cartão de crédito.
- Aplicações **espaciais**, que armazenam a localização espacial de dados, como informações de clima, mapas usados em sistemas de informações geográficas e em sistemas de navegação de automóveis.
- Aplicações de **série temporais**, que armazenam informações como dados econômicos em pontos regulares no tempo, como vendas diárias e valores mensais do Produto Interno Bruto (PIB).

Logo ficou aparente que os sistemas relacionais básicos não eram muito adequados para muitas dessas aplicações, em geral por um ou mais dos seguintes motivos:

- Estruturas de dados mais complexas eram necessárias para modelar a aplicação do que a representação relacional simples.
- Novos tipos de dados eram necessários além dos tipos básicos numéricos e alfanuméricos.
- Novas operações e construtores de linguagem de consulta eram necessários para manipular os novos tipos de dados.
- Novas estruturas de armazenamento e indexação eram necessárias para a pesquisa eficiente sobre os novos tipos de dados.

Isso levou os desenvolvedores de SGBD a acrescentarem funcionalidade a seus sistemas. Alguma funcionalidade era de uso geral, como a incorporação de conceitos dos bancos de dados orientados a objeto aos sistemas relacionais. Outras eram de uso especial, na forma de módulos opcionais que poderiam ser usados para aplicações específicas. Por exemplo, os usuários poderiam comprar um módulo de séries temporal para usar com seu SGBD relacional para aplicações de séries temporais.

Muitas organizações de grande porte utilizam vários pacotes de aplicação de software que trabalham intimamente com o **banco de dados de back-ends**. O banco de dados do back-end. representa um ou mais bancos de dados, possivelmente de diferentes fornecedores e usando diferentes modelos de dados, que mantêm dados manipulados por esses pacotes para dar suporte a transações, gerar relatórios e responder a consultas ocasionais. Um dos sistemas mais utilizados inclui o **ERP** (Enterprise Resource Planning, planejamento de recursos empresariais), que serve para consolidar diversas áreas funcionais dentro de uma organização, incluindo produção, vendas, distribuição, marketing, finanças, recursos humanos, e assim por diante. Outro tipo de sistema popular é o **CRM** (Customer Relationship Management, gerenciamento do relacionamento com o cliente), que compreende funções de processamento de pedido, bem como marketing e suporte ao cliente. Essas aplicações são habilitadas para Web porque usuários internos e externos recebem uma série de interfaces de portal Web para interagir com os bancos de dados de back-end.

1.7.6 Bancos de dados *versus* recuperação de informações

Tradicionalmente, a tecnologia de banco de dados se aplica a dados estruturados e formatados, que surgem em aplicações de rotina no governo, no comércio e na indústria. Ela é bastante utilizada nos setores de manufatura, varejo, bancos, seguros, finanças e saúde, onde dados estruturados são coletados por meio de formulários, como faturas ou documentos de registro de paciente. Uma área relacionada à tecnologia de banco de dados é a **Recuperação de Informação** (**RI**), que lida com livros, manuscritos e diversas formas de artigos baseados em biblioteca. O dado é indexado, catalogado e anotado usando palavras-chave. A RI está relacionada à busca por conteúdo com base nessas palavras-chave e a muitos problemas que lidam com processamento de documento e processamento de texto em forma livre. Muito trabalho tem sido feito sobre busca em texto baseada em palavras-chave, localização de documentos e sua classificação conforme a relevância, categorização automática de texto, classificação de documentos de texto por tópicos, e assim por diante. Com o advento da Web e a proliferação de páginas HTML na faixa dos bilhões, é preciso aplicar muitas técnicas de RI para processar os dados na Web. Os dados dessas páginas normalmente contêm imagens, texto e objetos que são ativos e mudam de maneira dinâmica. A recuperação de informações na Web é um problema novo que exige que técnicas de bancos de dados e RI sejam aplicadas a uma série de combinações novas. Discutiremos os conceitos relacionados à recuperação de informações e páginas Web no Capítulo 27.

1.8 Quando não usar um SGBD

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

- Alto investimento inicial em hardware, software e treinamento.
- A generalidade que um SGBD oferece para a definição e o processamento de dados.
- Esforço adicional para oferecer funções de segurança, controle de concorrência, recuperação e integridade.

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

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

Certos setores e aplicações decidiram não utilizar SGBDs de uso geral. Por exemplo, muitas ferramentas de projeto auxiliado por computador (CAD) usadas por engenheiros civis e mecânicos possuem software proprietário para gerenciamento de arquivos e dados, preparado para as manipulações internas dos desenhos e objetos 3D. De modo semelhante, sistemas de comunicação e comutação projetados por empresas como a AT&T foram manifestações

iniciais do software de banco de dados preparado para executar de forma muito rápida com dados organizados hierarquicamente, para agilizar o acesso e o roteamento das chamadas. De modo semelhante, implementações dos sistemas de informações geográficas (SIG) normalmente usam os próprios esquemas de organização de dados, a fim de implantar, de modo eficiente, funções relacionadas a processamento de mapas, contornos físicos, linhas, polígonos, e assim por diante. Os SGBDs de uso geral são inadequados para essa finalidade.

Resumo

Neste capítulo, definimos um banco de dados como uma coleção de dados relacionados, onde *dados* significam fatos gravados. Um banco de dados típico representa algum aspecto do mundo real e é usado para fins específicos por um ou mais grupos de usuários. Um SGBD é um pacote de software generalizado para implementar e manter um banco de dados computadorizado. Juntos, o banco de dados e o software formam um sistema de banco de dados. Identificamos várias características que distinguem a técnica de banco de dados das aplicações tradicionais de processamento de arquivo, e discutimos as principais categorias de usuários de banco de dados, ou os *atores em cena*. Observamos que, além dos usuários em ambiente de banco de dados, existem várias categorias de pessoal de suporte, ou *trabalhadores de bastidores*, em um ambiente de banco de dados.

Apresentamos uma lista de capacidades que devem ser fornecidas pelo software de SGBD ao DBA, aos projetistas de banco de dados e aos usuários finais para ajudá-los a projetar, administrar e usar um banco de dados. Depois, mostramos uma rápida perspectiva histórica da evolução das aplicações de banco de dados. Indicamos o casamento da tecnologia de banco de dados com a tecnologia da recuperação de informações, que desempenhará um papel importante devido à popularidade da Web. Finalmente, discutimos os custos adicionais do uso de um SGBD e algumas situações em que sua utilização pode não ser vantajosa.

Perguntas de revisão

1.1. Defina os seguintes termos: *dados*, *banco de dados*, *SGBD*, *sistema de banco de dados*, *catálogo de banco de dados*, *independência entre dados e programas*, *visão do usuário*, *DBA*, *usuário final*, *transação programada*, *sistema de banco de dados dedutivo*, *objeto persistente*, *metadados* e *aplicação para processamento de transação*.

1.2. Quais os quatro tipos principais de ações que envolvem bancos de dados? Discuta cada tipo rapidamente.

1.3. Discuta as principais características da abordagem de banco de dados e como ela difere dos sistemas de arquivo tradicionais.

1.4. Quais são as responsabilidades do DBA e dos projetistas de banco de dados?

1.5. Quais são os diferentes tipos de usuários finais de banco de dados? Discuta as principais atividades de cada um.

1.6. Discuta as capacidades que devem ser fornecidas por um SGBD.

1.7. Discuta as diferenças entre sistemas de banco de dados e sistemas de recuperação de informações.

Exercícios

1.8. Identifique algumas operações informais de consulta e atualização que você esperaria aplicar ao banco de dados mostrado na Figura 1.2.

1.9. Qual é a diferença entre redundância controlada e não controlada? Dê exemplos.

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

1.11. Mostre algumas visões adicionais que podem ser necessárias a outros grupos de usuários do banco de dados mostrado na Figura 1.2.

1.12. Cite alguns exemplos de restrições de integridade que você acredita que possam se aplicar ao banco de dados mostrado na Figura 1.2.

1.13. Dê exemplos de sistemas em que pode fazer sentido usar o processamento de arquivos tradicional em vez da técnica de banco de dados.

1.14. Considere a Figura 1.2.

- **a.** Se o nome do departamento 'CC' (Ciência da computação) mudar para 'CCES' (Ciência da computação e engenharia de software) e o prefixo correspondente para o número da disciplina também mudar, identifique as colunas no banco de dados que precisariam ser atualizadas.
- **b.** Você consegue reestruturar as colunas nas tabelas DISCIPLINA, TURMA e PRE_REQUISITO de modo que somente uma delas precise de atualização?

Bibliografia selecionada

A edição de outubro de 1991 de *Communications of the ACM* e Kim (1995) incluem vários artigos que descrevem SGBDs da próxima geração. Muitos dos recursos de banco de dados discutidos no início agora estão disponíveis comercialmente. A edição de março de 1976 de *ACM Computing Surveys* oferece uma introdução aos sistemas de banco de dados, e pode fornecer uma perspectiva histórica para o leitor interessado.