Capítulo

1

Introdução

Este capítulo apresenta os conceitos básicos da área de banco de dados que são necessário à compreensão do projeto de banco de dados. São apresentados conceitos como banco de dados, sistema de gerência de banco de dados e modelo de dados. Além disso é fornecida uma visão geral do projeto de banco de dados

O leitores que já conhece os fundamentos de banco de dados provavelmente poderá passar diretamente ao próximo capítulo.

1.1 BANCO DE DADOS

1.1.1 Compartilhamento de dados

Muitas vezes, a implantação da Informática em organizações ocorre de forma evolutiva e gradual. Inicialmente, apenas determinadas funções são automatizadas. Mais tarde, à medida que o uso da Informática vai se estabelecendo, novas funções vão sendo informatizadas.

Para exemplificar, vamos considerar uma indústria hipotética. Consideramos que nesta indústria são executadas três funções:

- *Vendas*

  Esta função concentra as atividades da indústria relativas ao contato com os clientes, como fornecimento de cotações de preços, vendas, e informações sobre disponibilidade de produtos.

- *Produção*

  Esta função concentra as atividades da indústria relativas à produção propriamente dita, como planejamento da produção e controle do que foi produzido.

- *Compras*

  Esta função concentra as atividades da indústria relativas à aquisição dos insumos necessários à produção, como cotações de preços junto a fornecedores, compras e acompanhamento do fornecimento.

No exemplo mencionado acima, os dados de um produto são usados em várias funções. Estes dados são necessários no planejamento de produção, pois para planejar o que vai ser produzido, é necessário conhecer como os produtos são estruturados (quais seus componentes) e como são produzidos. Os dados de produto também são necessários no setor de compras, pois este necessita saber que componentes devem ser adquiridos. Já o setor de vendas também necessita conhecer dados de produtos, como por exemplo seu preço, seu estoque atual, seu prazo de fabricação, etc.

Se cada uma das funções acima for informatizada de forma separada, sem considerar a informatização das demais funções, pode ocorrer que, para cada uma das funções, seja criado um arquivo separado de produtos (ver Figura 1.1).

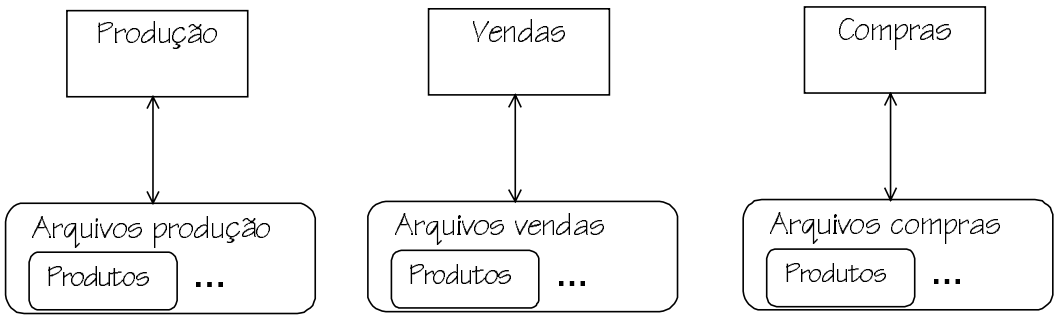

Figura 1.1: Sistemas isolados

Neste caso, surge o problema da *redundância de dados*. Redundância de dados ocorre quando uma determinada informação está representada no sistema em computador várias vezes. No caso do exemplo, estão redundantes as informações referentes a um produto, que aparecem nos arquivos de produtos de cada um dos três sistemas.

Há duas formas de redundância de dados, a redundância *controlada* de dados e a redundância *não controlada* de dados.

A redundância *controlada* de dados acontece quando o software tem conhecimento da múltipla representação da informação e garante a sincronia entre as diversas representações. Do ponto de vista do usuário externo ao sistema em computador, tudo acontece como se existisse uma única representação da informação. Essa forma de redundância é utilizada para melhorar a performance global do sistema. Um exemplo é um sistema distribuído, onde uma mesma informação é armazenada em vários computadores, permitindo acesso rápido a partir de qualquer um deles.

A redundância *não controlada* de dados acontece quando a responsabilidade pela manutenção da sincronia entre as diversas representações de uma informação está com o usuário e não com o software. Este tipo de redundância deve ser evitado, pois traz consigo vários tipos de problemas:

- *Redigitação*

  A mesma informação é digitada várias vezes. No caso do exemplo da indústria, os dados de um produto são digitados no setor de vendas, no setor de produção e no setor de compras. Além de exigir trabalho desnecessário, a redigitação pode resultar em erros de transcrição de dados.

- *Inconsistências de dados*

  A responsabilidade por manter a sincronia entre as informações é do usuário. Por erro de operação, pode ocorrer que uma representação de uma informação seja modificada, sem que as demais representações o sejam. Exemplificando, uma alteração na estrutura de um determinado produto pode ser informada através do sistema de produção e deixar de ser informada nos demais sistemas. A estrutura do produto passa a aparecer de forma diferente nos vários sistemas. O banco de dados passa a ter informações *inconsistentes*.

A solução para evitar a redundância não controlada de informações é o *compartilhamento de dados*. Nesta forma de processamento, cada informação é armazenada uma única vez, sendo acessada pelos vários sistemas que dela necessitam (Figura 1.2). Ao conjunto de arquivos integrados que atendem a um conjunto de sistemas dá-se o nome de *banco de dados (BD)*.

banco de dados = conjunto de dados integrados que tem por objetivo atender a uma comunidade de usuários

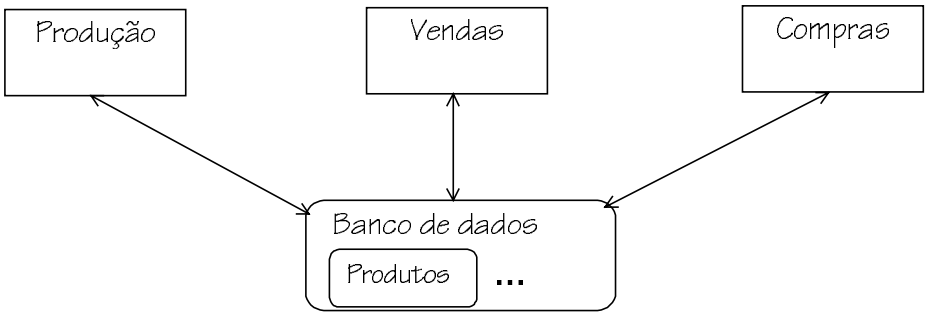

Figura 1.2: Sistemas integrados com dados compartilhados

O compartilhamento de dados tem reflexos na estrutura do software. A estrutura interna dos arquivos passa a ser mais complexa, pois estes devem ser construídos de forma a atender às necessidades dos diferentes sistemas. Para contornar este problema, usa-se um *sistema de gerência de banco de dados*, conforme descrito na próxima seção.

1.1.2 Sistema de Gerência de Banco de Dados

A programação de aplicações em computadores sofreu profundas modificações desde seus primórdios. No início, quando usavam-se linguagens como COBOL, Basic, C e outras, os programadores incorporavam em um programa toda funcionalidade desejada. O programa continha as operações da interface de usuário, as transformações de dados e cálculos, as operações de armazenamento de dados, bem como as tarefas de comunicação com outras sistemas e programas.

Com o tempo, foram sendo identificadas funcionalidades comuns a muitos programas. Por exemplo, hoje em dia, a grande maioria dos programas comunica-se com os usuários através de interfaces gráficas de janelas. Entretanto, normalmente, os programas não contém todo código referente a exibição dos dados na interface, mas utilizam *gerenciadores de interface de usuário*, conjuntos de rotinas que incluem as funcionalidades que um programador vai necessitar freqüentemente, ao construir uma interface de usuário. Da mesma forma, para comunicar-se com processos remotos, usam *gerenciadores de comunicação*. Para manter grandes repositórios compartilhados de dados, ou seja, para manter *bancos de dados*, são usados *sistemas de gerência de banco de dados* (SGBD).

sistema de gerência de banco de dados
=
software que incorpora as funções de definição, recuperação e alteração de dados em um banco de dados

Essa modularização de programas tem várias vantagens. A *manutenção* de programas torna-se mais simples, pois uma separação clara de funções torna programas mais facilmente compreensíveis. A *produtividade* de progra-

madores também aumenta, já que os programas ficam menores, pois usam funções já construídas.

No mercado, há vários tipos de SGBD. Neste livro, nos concentramos em um tipo de SGBD, o *relacional*, que domina o mercado da atualidade. Entretanto, muitas das idéias apresentadas nas seções referentes à modelagem de dados aplicam-se também a outros tipos, como os SGBD orientados a objetos ou objeto/relacionais.

1.2 Modelos de Banco de Dados

Um *modelo de (banco de) dados* é uma descrição dos tipos de informações que estão armazenadas em um banco de dados. Por exemplo, no caso da indústria citado acima, o modelo de dados poderia informar que o banco de dados armazena informações sobre produtos e que, para cada produto, são armazenados seu código, preço e descrição. Observe que o modelo de dados não informa quais os produtos que estão armazenados no banco de dados, mas apenas que o banco de dados contém informações sobre produtos.

modelo de dados = descrição formal da estrutura de um banco de dados

Para construir um modelo de dados, usa-se uma *linguagem de modelagem de dados*. Linguagens de modelagem de dados podem ser classificadas de acordo com a forma de apresentar modelos, em linguagens *textuais* ou linguagens *gráficas*. Como veremos adiante, um mesmo modelo de dados pode ser apresentado de várias formas. Cada apresentação do modelo recebe a denominação *esquema de banco de dados*.

De acordo com a intenção do modelador, um banco de dados pode ser modelado (descrito) há vários níveis de abstração. Um modelo de dados que servirá para explicar a um usuário qual é a organização de um banco de dados provavelmente não conterá detalhes sobre a representação em meio físico das informações. Já um modelo de dados usado por um técnico para otimizar a performance de acesso ao banco de dados conterá mais detalhes de como as informações estão organizadas internamente e portanto será menos abstrato.

No projeto de banco de dados, normalmente são considerados dois níveis de abstração de modelo de dados, o do *modelo conceitual* e o do *modelo lógico*.

1.2.1 Modelo conceitual

Um *modelo conceitual* é uma descrição do banco de dados de forma independente de implementação em um SGBD. O modelo conceitual registra que dados podem aparecer no banco de dados, mas não registra como estes dados estão armazenados a nível de SGBD.

modelo conceitual = modelo de dados abstrato, que descreve a estrutura de um banco de dados de forma independente de um SGBD particular

A técnica mais difundida de modelagem conceitual é a *abordagem entidade-relacionamento* (ER). Nesta técnica, um modelo conceitual é usualmente representado através de um diagrama, chamado *diagrama entidade-relacionamento (DER)*. A Figura 1.3 apresenta um DER parcial para o problema da fábrica.

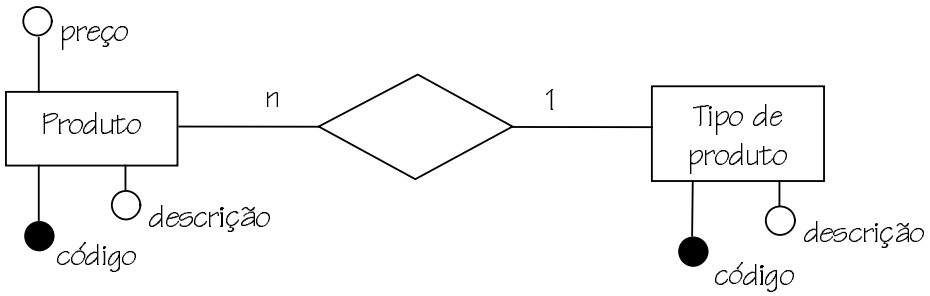

Figura 1.3: Exemplo de modelo conceitual

Entre outras coisas, este modelo informa que o banco de dados contém dados sobre produtos e sobre tipos de produtos. Para cada produto, o banco de dados armazena o código, a descrição, o preço, bem como o tipo de produto ao qual está associado. Para cada tipo de produto, o banco de dados armazena o código, a descrição, bem como os produtos daquele tipo.

1.2.2 Modelo lógico

Um *modelo lógico* é uma descrição de um banco de dados no nível de abstração visto pelo usuário do SGBD. Assim, o modelo lógico é dependente do tipo particular de SGBD que está sendo usado.

No presente livro, serão tratados apenas modelos lógicos referentes a SGBD relacional. Em um SGBD relacional, os dados estão organizados na forma de tabelas.

A Figura 1.4 mostra um exemplo de BD relacional projetado a partir do modelo conceitual mostrado na Figura 1.3. Um modelo lógico para o BD acima deve definir quais as tabelas que o banco contém e, para cada tabela, quais os nomes das colunas.

Abaixo é apresentado o modelo lógico (de forma textual) da Figura 1.4:

TipoDeProduto(CodTipoProd,DescrTipoProd)
Produto(CodProd,DescrProd,PrecoProd,CodTipoProd)
CodTipoProd referencia TipoDeProduto

TipoDeProduto

CodTipoProd	DescrTipoProd
1	Computador
2	Impressora

Produto

CodProd	DescrProd	PrecoProd	CodTipoProd
1	PC desktop modelo X	2.500	1
2	PC notebook ABC	3.500	1
3	Impressora jato de tinta	600	2
4	Impressora laser	800	2

Figura 1.4: Exemplo de tabelas de BD relacional

O modelo lógico descreve a estrutura do banco de dados, conforme vista pelo usuário do SGBD. Detalhes de armazenamento interno de informações, que não tem influencia sobre a programação de aplicações no SGBD, mas podem influenciar a performance da aplicações (por exemplo, as estruturas de arquivos usadas no acesso as informações) não fazem parte do modelo lógico. Estas são representadas no *modelo físico*. Modelos físicos não são tratados neste livro. Eles são usados apenas por profissionais que fazem *sintonia* de banco de dados, procurando otimizar a performance. As linguagens e notações para o modelo físico não são padronizadas e variam de produto a produto. A tendência em produtos mais modernos é esconder o modelo físico do usuário e transferir a tarefa de otimização ao próprio SGBD.

modelo lógico = modelo de dados que representa a estrutura de dados de um banco de dados conforme vista pelo usuário do SGBD

1.2.3 Modelo conceitual como modelo de organização

Quando se observa um conjunto de arquivos em computador, sejam eles gerenciados por um SGBD, sejam eles arquivos convencionais, verifica-se que usualmente um arquivo contém informações sobre um conjunto de objetos ou *entidades* da organização que é atendida pelo sistema em computador. Assim, no exemplo da indústria acima citado, um sistema em computador provavelmente conteria um arquivo para armazenar dados de produtos, outro para armazenar dados de vendas, outro para armazenar dados de ordens de compra e assim por diante.

Desta constatação surgiu uma das idéias fundamentais do projeto de banco de dados: a de que através da identificação das entidades que terão informações representadas no banco de dados, é possível identificar os arquivos que comporão o banco de dados. Devido a esta relação um-para-um entre ar-

quivos em computador e entidades da organização modelada, observou-se que um mesmo modelo conceitual pode ser usado em duas funções:

- como *modelo abstrato da organização*, que define as entidades da organização que tem informações armazenadas no banco de dados, e
- como *modelo abstrato do banco de dados*, que define que arquivos farão parte do banco de dados.

Exemplificando, se considerarmos o modelo da Figura 1.3 podemos interpretá-lo de duas formas. Em uma interpretação, como modelo abstrato da organização, o diagrama nos informa que na organização há produtos e tipos de produtos, que associado a cada tipo de produto há um código do tipo e uma descrição e assim por diante. Na outra interpretação, como modelo abstrato de um banco de dados, o diagrama nos informa que o banco de dados deverá conter informações sobre produtos e tipos de produtos, que para cada tipo de produto são armazenados seus código e sua descrição e assim por diante.

Na prática, convencionou-se iniciar o processo de construção de um novo banco de dados com a construção de um modelo dos objetos da organização que será atendida pelo banco de dados, ao invés de partir diretamente para o projeto do banco de dados.

Esta forma de proceder permite envolver o usuário na especificação do banco de dados. Sabe-se da prática da engenharia de software que o envolvimento do usuário na especificação do software aumenta a qualidade do software produzido. A idéia é que o usuário é aquele que melhor conhece a organização e, portanto, aquele que melhor conhece os requisitos que o software deve preencher. Modelos conceituais são modelos que descrevem a organização e portanto são mais simples de compreender por usuários leigos em Informática, que modelos que envolvem detalhes de implementação.

1.3 Projeto de BD

O projeto de um novo BD dá-se em duas fases:

1 *Modelagem conceitual*

Nesta primeira fase, é construído um modelo conceitual, na forma de um diagrama entidade-relacionamento. Este modelo captura as necessidades da organização em termos de armazenamento de dados de forma independente de implementação.

2 *Projeto lógico*

A etapa de projeto lógico objetiva transformar o modelo conceitual obtido na primeira fase em um modelo lógico. O modelo lógico define como o banco de dados será implementado em um SGBD específico.

O processo acima é adequado para a construção de um novo banco de dados. Caso já exista um banco de dados ou um conjunto de arquivos convencionais, e se pretenda construir um novo banco de dados, o processo acima é modificado e incorpora uma etapa de *engenharia reversa* de banco de dados. A engenharia reversa é explicada nos capítulos 5 e 6.