Capítulo 2

Modelo de entidade e relacionamento

- Conceitos
- Normalização
- Fases de um projeto utilizando ER

A necessidade de organizar informações acompanha a humanidade desde o início dos tempos. O pastor representava ovelhas por meio de pedras, enquanto os escribas ordenavam os textos nas estantes. Acontece que a população mundial cresceu, assim como a quantidade de informações disponíveis, e, desse modo, pesquisá-las tornou-se mais complexo, o que exigiu novas formas de organização.

Tente achar um livro na biblioteca (figura11) de sua escola sem saber como estão organizadas as estantes. Vá procurando estante por estante, livro por livro... Quanto tempo irá demorar? Agora, pense em fazer a mesma pesquisa na Biblioteca Nacional (do Rio de Janeiro) ou na Biblioteca do Smithsonian Museum, dos Estados Unidos, que são muito maiores que a de sua escola. Sem uma organização prévia fica muito demorado e trabalhoso obter as informações de que precisamos em nosso dia a dia.

E como a informática pode ajudar na organização? Muito e em todo esse processo, que inclui armazenamento, manutenção e consulta de informações, proporcionando agilidade, uniformidade e segurança em todas as suas fases. Ao juntarmos o conhecimento preexistente à velocidade de processamento e à capacidade de armazenamento de informações que a informática oferece, chegamos a modelos extremamente interessantes no que diz respeito à facilidade de uso, velocidade de acesso e de respostas, além de baixo custo de manutenção.

Depois de vários estudos, chegou-se a uma metodologia para projetar e construir bancos de dados otimizados, capazes de permitir o acesso mais rápido e consistente às informações, utilizando espaço cada vez menor de armazenamento. É sobre essa metodologia que falaremos neste capítulo, o modelo de entidade e relacionamento, que propõe definições e regras para projeto e implementação de bancos de dados, assim como a relação desses dados com as funcionalidades do sistema.

As bases do modelo de entidade e relacionamento começaram a ser lançadas quando Edgar Frank Codd definiu as principais implementações necessárias para o correto funcionamento de um banco de dados relacional usando, para isso, a teoria dos conjuntos, a álgebra e o cálculo relacional. O modelo ganhou mais corpo quando Donald D. Chamberlin e Raymond F. Boyce desenvolveram uma linguagem de consulta que facilitava o acesso e a manutenção de bancos de dados relacionais, oferecendo os recursos necessários para sua utilização em larga escala, o que atendia às necessidades do mercado. A contribuição de Peter Chen foi na definição de uma metodologia para modelagem de projetos de banco de dados, utilizando banco de dados relacionais (veja quadro *Os nomes por trás da tecnologia*).

A linguagem criada por Clamberlin e Boyce ganhou o nome de SQL, e somente em 1986 foi distribuída e aceita por praticamente todos os bancos de dados, tor-

Figura 11
Se soubermos como estão organizadas as estantes, podemos encontrar um livro facilmente, seja qual for o tamanho de uma biblioteca.

Figura 12
Não é possível encontrar um só livro em ambiente desorganizado.

ROBERT W. GINN/ALAMY/OTHER IMAGES

nando-se referência, com o lançamento do SQL padrão ANSI. A padronização foi necessária porque cada banco de dados, por questão de projeto ou facilidade de implementação, modificava os comandos da linguagem, a tal ponto que, hoje, se não fosse a padronização, provavelmente teríamos de reaprendê-la a cada mudança de sistema gerenciador de banco de dados. A linguagem SQL será um dos focos do terceiro capítulo deste livro.

Infelizmente, a padronização ainda não gerou uma linguagem com funções totalmente iguais, o que nos obriga, ao trocarmos de sistema gerenciador de banco de dados, a pesquisar, por exemplo, a função que retorna à data atual no SQL. Mas, com certeza, as diferenças entre elas são atualmente bem menores. Hoje em dia, o padrão ANSI está na versão SQL:2003.

Os estudos não pararam por aí. O modelo de entidade e relacionamento foi inegavelmente um marco na história da informática, utilizado em larga escala, mas avançou-se bastante depois dele. Hoje, por exemplo, existe a UML (Linguagem de Modelagem Unificada), outra técnica de modelagem, baseada na teoria de Orientação a Objetos (analisada no capítulo 4).

2.1. Conceitos

Para podermos utilizar as técnicas do modelo de entidade e relacionamento, necessitamos predefinir alguns de seus conceitos, de modo a facilitar seu entendimento.

Banco de dados

É um conjunto de informações inter-relacionadas sobre determinado assunto e armazenadas de forma a permitir acesso organizado por parte do usuário.

Bancos de dados relacional

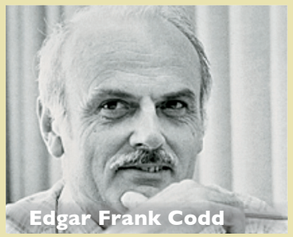
São conjuntos de dados, relacionados entre si, que implementam as características do modelo de entidade e relacionamento.

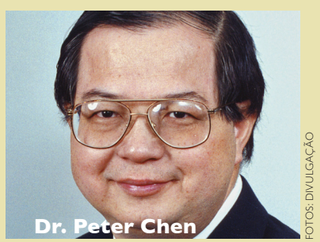
Sistema gerenciador de bancos de dados (SGBD)

É um conjunto de programas que permite a implementação de bancos de dados,

Os nomes por trás da tecnologia

Edgar Frank Codd, Donald D. Chamberlin e Peter Pin Shan Chen são os pesquisadores que mais contribuíram para a criação do modelo de entidade e relacionamento. Em junho de 1970, Codd, matemático inglês, que na época trabalhava no laboratório da IBM em San José, na Califórnia, Estados Unidos, publicou um artigo decisivo na revista ACM – Association for Computer Machinery (Associação para Maquinária da Computação), intitulado *Relational Model of Data for Large Shared Data Banks* (*Modelo de dados relacional para grandes bancos de dados compartilhados*). Quatro anos depois, em maio de 1974, Chamberlin e Raymond F. Boyce, ambos engenheiros e cientistas da IBM, apresentaram o trabalho *SEQUEL – A Structured English Query Language* (*Linguagem de consulta estruturada em inglês*). Em março de 1976, Peter Chen, engenheiro elétrico e Ph.D. em Ciência da Computação, publicou, também na revista ACM, o artigo *The Entity-Relationship Model-Toward a Unified View of Data* (*O modelo de entidade e relacionamento, uma visão unificada dos dados*).





FOTOS: DIVULGAÇÃO

assim como o controle de acesso, o backup, a recuperação de falhas, a manutenção da integridade, a administração e a segurança dos dados que contém.

Para a construção de modelos, é preciso abstrair, isto é, não se preocupar com todos os detalhes, mas apenas com os que se quer analisar e/ou sobre os quais se tem alguma dúvida.

Modelo

Podemos definir um modelo como sendo um protótipo, em escala menor, do produto que queremos implementar ou da solução que queremos obter.
O **modelo** nos permite, com um custo muito menor (de tempo, dinheiro e trabalho), em comparação ao do desenvolvimento do produto final, analisar e desenvolver alguns aspectos que farão a diferença no produto final. Ou seja, no modelo podemos criar, testar funcionalidades novas e avaliar o projeto, com um baixo custo antes de sua implementação.

Abstração

Para exemplificar o que é abstração, vale acompanhar um exemplo bem corriqueiro e que muita gente vivencia. Quando olhamos para uma casa, podemos pensar em seu tamanho, sua localização, no número de quartos que a integram, na cor das paredes. Já um engenheiro civil, ao olhar para a mesma casa, pensará em como ela foi construída, se o material utilizado é de boa qualidade, se sua estrutura foi concebida para suportar seu tamanho. O pedreiro pensará na quantidade de tijolos, cimento, pedras, areia e ferro que foram necessários para construí-la. O jardineiro avaliará sua localização, o clima e sua posição em relação ao sol, além das melhores plantas para o jardim. O encanador pode refletir sobre qual é o tamanho da caixa d'água para garantir o abastecimento na casa e em quantos metros de encanamento de cada largura foram necessários para seu sistema hidráulico. O eletricista talvez pense sobre a metragem e a bitola dos fios empregados no sistema elétrico. O corretor de imóveis se concentra na metragem da casa, no número de cômodos, de vagas na garagem, na localização, no preço de aluguel ou venda (figura 13).

ATENÇÃO

O Modelo ER e os Sistemas Gerenciadores de Bancos de Dados foram criados primeiramente para aceitar nomes em inglês, língua que não possui acentos em suas palavras. Apesar de hoje em dia a maioria dos SGBD´s aceitarem palavras acentuadas e até o uso do caracter espaço entre as palavras que nomeiam algum de seus componentes, por convensão, não utilizaremos espaços nem acentos para nominar os componentes de nossos modelos afim de não gerarmos problemas de implementação em qualquer que seja o SGBD, padronizarermos a implementação, evitando dúvidas do tipo Funcionário com ou sem acento, sem falar nas mudanças da língua.

Cada um dos personagens do exemplo se fixou em detalhes diferentes sobre um mesmo projeto, a casa. Olhou para ela pensando em suprir as próprias necessidades, enfatizando as características mais importantes para atendê-las, e desprezou os demais detalhes, os quais, embora também façam parte da construção, não têm relevância particular.

É o que se chama de abstração: esses vários pontos de vista, essas diferentes possiblidades de análises sobre um mesmo objeto é o que se chama de abstração, uma característica fundamental para a construção de um bom modelo de entidade e relacionamento.

Modelo de entidade e relacionamento

Propõe definições e regras para o projeto e a implementação de bancos de dados, assim como a relação desses dados com as funcionalidades que esse sistema deve implementar. Sugere que, nas diversas fases de desenvolvimento do projeto, os modelos sejam refinados até que se chegue ao modelo final que, em modelagem ER, chamamos de projeto físico. Acrescentando-se a seguir o projeto físico do banco de dados, ele se juntará com as funcionalidades dos programas da aplicação, para só então chegar-se a uma solução completa de software. Há vários componentes do modelo ER. Vale, portanto, conhecer os principais, entre os quais se alinham: Entidade, Relacionamento, Atributo e Chaves.

Figura 13
Vários olhares sobre um mesmo tema.

Entidades

Entidades são abstrações do mundo real que contêm um conjunto de informações inter-relacionadas e coerentes. Estas informações são chamadas de atributos. Toda entidade possui um nome que a identifica, geralmente formado por um substantivo no singular. A representação gráfica de uma entidade é feita por um retângulo com seu nome no centro, como mostra a figura 14.

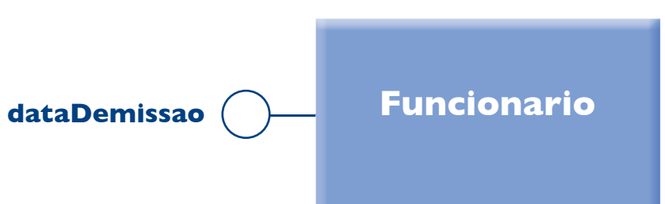
Figura 14
Entidade Funcionario.

Atributo

Atributo é cada informação que compõe uma entidade. Possui um nome, um tipo e um tamanho (número de caracteres). De modo genérico o tipo, pode ser nominado como texto, número, data, hora, etc. até que se saiba em qual sistema gerenciador de banco de dados este será implementado e então se atribua o tipo correto, pois cada SGBD possui suas particularidades em relação aos tipos de dados aceitos. Por exemplo os tipos real ou double.

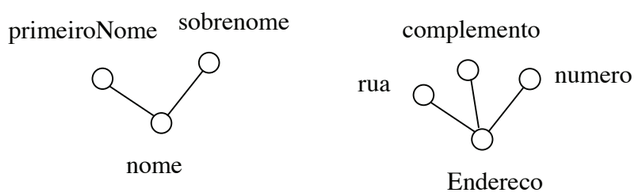
O atributo pode ser representado no diagrama ER como um círculo, com o nome ao lado ou como uma elipse com seu nome, o qual é representado geralmente por um substantivo. Para evitar problemas de compatibilidade, deve começar com uma letra e não conter espaços e acentuação, mas pode incluir caracteres especiais como underline, entre outros (figura 15).

Figura 15
Atributo.



Há alguns tipos de atributos especiais usados para demonstrar a estrutura das informações que eles representam – de modo a facilitar a busca dessas informações – ou o relacionamento entre as entidades. São eles:

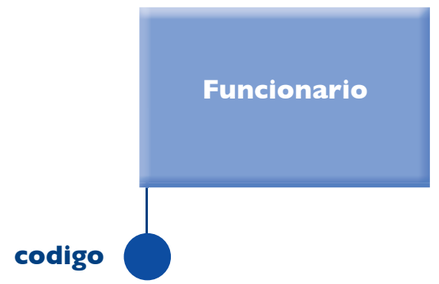
1. **Atributo composto:** representa a estrutura das informações que serão armazenadas no atributo, por exemplo:



2. **Atributo multivalorado:** pode receber mais de um valor ao mesmo tempo. Um bom exemplo é o atributo habilidades de um funcionário, que será preenchido com a lista de suas aptidões separadas por vírgulas. Veja um exemplo de preenchimento: liderança, boa comunicação, bom relacionamento interpessoal. Assim, o atributo habilidades é considerado um atributo multivalorado.

3. **Chave primária:** atributo ou conjunto de atributos que identifica unicamente uma tupla (registro) em uma entidade. É expresso com um círculo preenchido, como mostra a figura 16.

Figura 16
Chave primária.



4. **Chave estrangeira:** atributo que implementa o relacionamento entre entidades e permite o controle da integridade referencial, isto é, é um atributo que, fazendo parte da chave primária em uma entidade, é incluído em outra entidade ou relacionamento, implementando as ligações entre elas.

Relacionamento

É o elemento responsável por definir as características das ligações entre as entidades. Representado graficamente por um losango, seu nome é em geral expresso por um verbo ou uma locução verbal. Por exemplo a figura 17.

Figura 17
Relacionamento.

Auto-relacionamento: indica um relacionamento entre as ocorrências de um mesmo relacionamento. Para demonstrar melhor do que se trata, vale definir os papéis de cada um de seus lados, como mostra a figura 18.

Figura 18
Auto-relacionamento.

Cardinalidade

Serve para definir o tipo de relacionamento entre as entidades. Existem duas notações para identificar a cardinalidade. Uma delas refere-se simplesmente ao valor máximo que a cardinalidade daquele relacionamento pode alcançar, e é grafada com o número 1 (que representa 1 elemento da entidade) ou com a letra N (que representa muitos ou mais de um elemento da entidade). A outra expressa o número mínimo e o número máximo de ocorrências em um relacionamento. Neste caso, sua notação é (1:N), onde 1 representa o número mínimo e N o número máximo de ocorrências. São quatro as cardinalidades:

1. Relacionamentos 1 para N

Indicam que uma ocorrência na entidade A pode estar relacionada a N ocorrências da entidade B. No exemplo da figura 19 podemos verificar que um vendedor atende N clientes e que um cliente é atendido por apenas 1 vendedor.

Figura 19
Relacionamento 1 para N.

2. Relacionamentos N para 1

Indicam que uma ocorrência na entidade B pode estar relacionada com N ocorrências na entidade A. No exemplo da figura 20, a 1 departamento podem pertencer N funcionários.

Figura 20
Relacionamento N para 1.

3. Relacionamentos N para N

Indicam que uma ocorrência na entidade A pode estar relacionada a N ocorrências na entidade B e que uma ocorrência na entidade B pode estar relacionada a N ocorrências na entidade A. No exemplo da figura 21 podemos observar que um cliente compra N produtos e que um produto pode ser comprado por N clientes.

Figura 21
Relacionamento N para N.

O relacionamento N para N possui uma característica especial. Também chamado de relacionamento com campos, sua implementação exige a inclusão de atributos (pelo menos na chave primária de cada uma das entidades envolvidas) e chave primária. Quando acontece a implementação do modelo físico, este relacionamento se torna uma tabela, como mostra a figura 22.

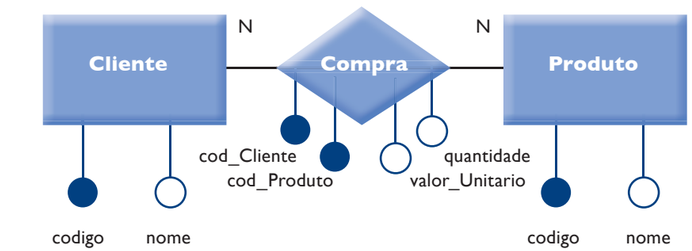


Figura 22
Relacionamento com campos.

Resumindo:

- Cliente tem os atributos “codigo” e “nome”, “codigo” é a chave primária.
- Produto também tem os atributos “codigo” e “nome”, tendo “codigo” como chave primária.
- Compra tem os atributos “cod_produto”, “cod_cliente”, que formam a chave primária, além dos atributos “quantidade” e “valor_unitario”.

4. Relacionamentos 1 para 1

Indicam que uma ocorrência da entidade A pode estar relacionada a uma ocorrência na entidade B e que uma ocorrência da entidade B pode estar relacionada a uma ocorrência da entidade A (figura 23).

Figura 23
Relacionamentos 1 para 1.

Restrição

Muitas vezes, simplesmente definir um relacionamento não nos garante total entendimento da situação que ele se deve demonstrar, conforme a figura 24.

Figura 24
Relacionamento que não nos garante entendimento da situação.

Esse exemplo nos parece claro quanto ao relacionamento entre funcionário e departamento, mas e se perguntarmos: pode haver funcionário sem departamento associado?

Esse questionamento, num primeiro momento, pode parecer sem sentido, mas imagine uma situação em que os funcionários passam por treinamento e só são alocados em departamentos depois de serem avaliados. E, então, eles não são funcionários? Claro que são! E é para deixar mais clara esta definição que existe o conceito de restrição, que expressa quais são o maior e o menor valor do relacionamento. Para o caso proposto a notação ficaria como na figura 25:

Figura 25
Restrição.

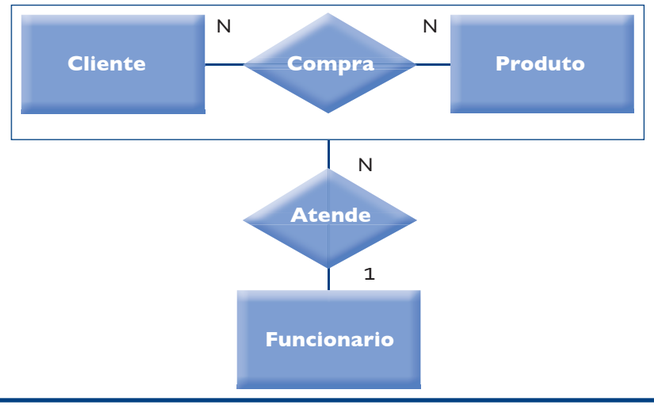
Como se deduz que um funcionário pode pertencer a no mínimo nenhum e no máximo 1 departamento e 1 departamento possui no mínimo 1 e no máximo N funcionários. Essa representação nos ajuda a definir as restrições de integridade de nosso modelo e permite maior compreensão do banco de dados a ser construído. O que devemos entender é que pode haver funcionários sem departamento, mas não departamentos sem funcionário.

Agregação

Outro problema conceitual que é preciso definir é o relacionamento de uma entidade com um conjunto de entidades, isto é, esse relacionamento só existe se houver um conjunto de informações.

Imagine a seguinte situação:
Um cliente compra produtos e é atendido por um funcionário. Veja na figura 26 como exibir graficamente esse caso:

Figura 26
Agregação.

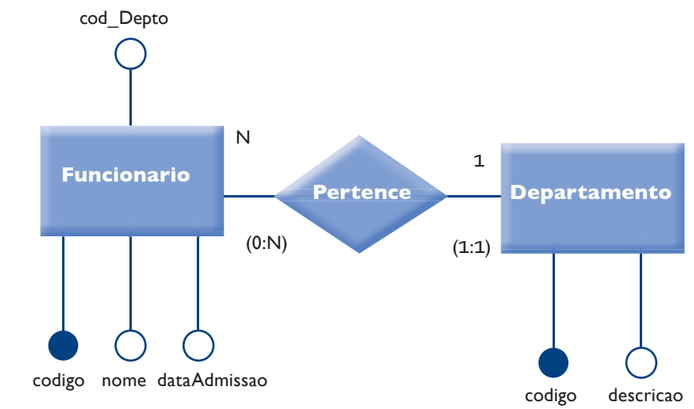


Interpretando o diagrama anterior, podemos dizer que um cliente compra N produtos e é atendido por 1 funcionário e que 1 funcionário atende N (clientes que compram produtos).

Diagrama de entidade e relacionamento

Parte do modelo de entidade e relacionamento, o diagrama é a representação gráfica dos elementos nele definidos. É montado após a denominação das entidades, dos seus atributos e relacionamentos (figura 27).

Figura 27
Diagrama de entidade e relacionamento.



É possível identificar nesse pequeno fragmento de diagrama quase todos os elementos do modelo visto até aqui:
Entidades: Funcionario e Departamento.
Relacionamento: Pertence.

Atributos:
- **da entidade Funcionario:** codigo, nome, dataAdmissao, cod_Depto.
- **da entidade Departamento:** codigo, descricao.

Chave primária:
- **da entidade Funcionario:** codigo.
- **da entidade Departamento:** codigo.

Chave estrangeira: o relacionamento Pertence com cardinalidade N para 1 faz com que seja necessário criar o campo cod_Depto na entidade Funcionario, que conterá o código do departamento a que cada funcionário pertence.

Restrições de integridade: podemos observar, pelas notações de restrição de integridade, que um funcionário tem de pertencer a um departamento. Já um departamento pode ser cadastrado sem um funcionário associado.

Vale, agora, propor um roteiro de passos para a elaboração do modelo de entidade e relacionamento e a criação do diagrama que o representará.

1. Listar as entidades candidatas a integrar o modelo.
2. Analisar e selecionar as entidades que realmente fazem parte do modelo, excluindo as demais.
3. Analisar os relacionamentos entre as entidades.
4. Definir a cardinalidade dos relacionamentos.
5. Definir os atributos das entidades e relacionamentos com campos e também as chaves primária e estrangeira (se houver).
6. Definir as restrições de integridade dos relacionamentos.
7. Desenhar o diagrama de entidade e relacionamento.

2.1.1. Estudo de caso

Vale imaginar como construir um modelo de entidade e relacionamento para o dono de uma pequena padaria, que será chamado de senhor João. No final, será elaborado o diagrama de entidade e relacionamento.

O senhor João vende, além de pães, vários outros tipos de produtos, tais como frios, laticínios, lanches, refrigerantes, sorvetes, balas, chicletes, chocolates, cartões telefônicos e artigos diversos expostos no balcão do caixa. Vende também, nos fins de semana, frango assado.

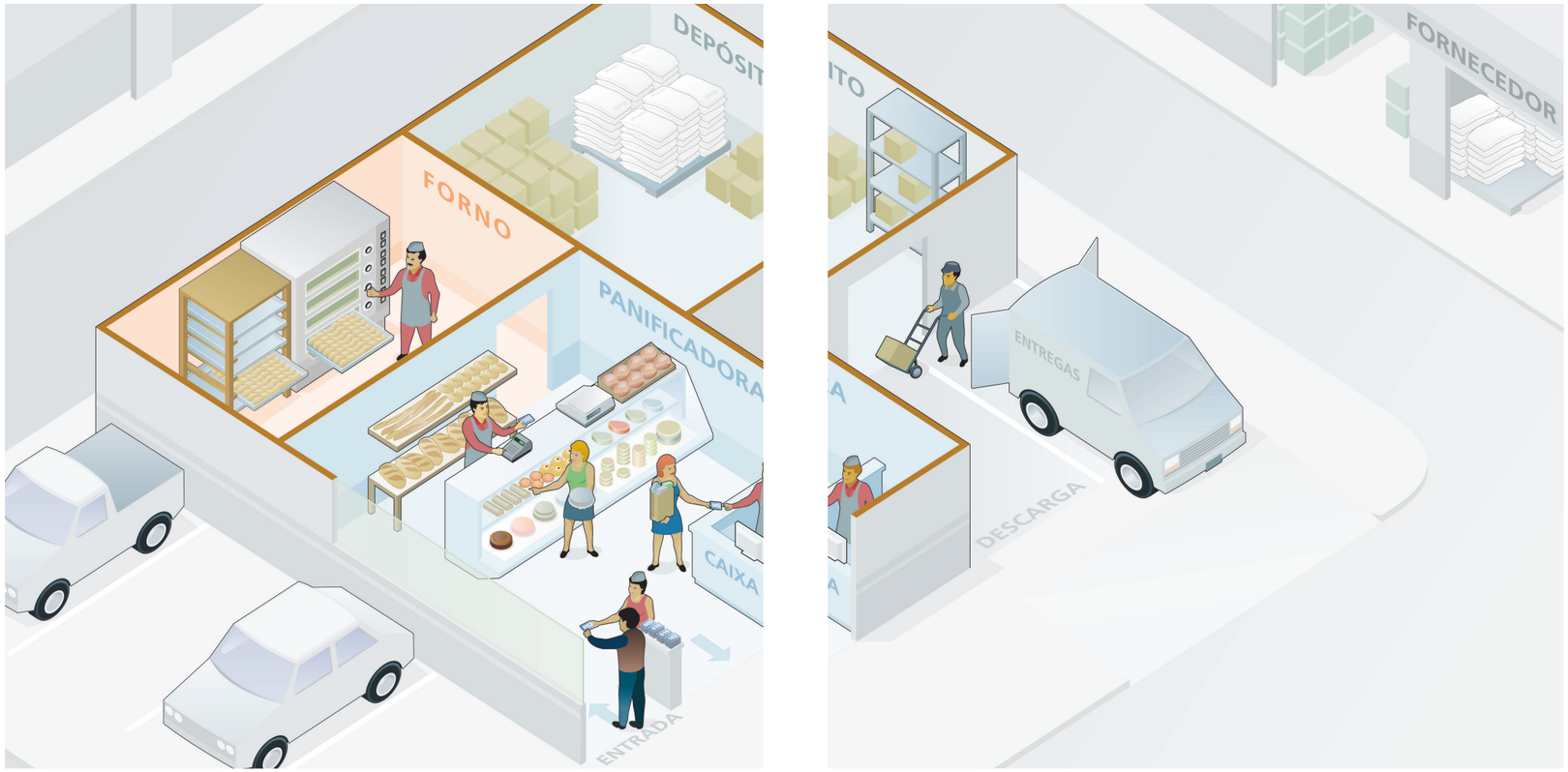
Na padaria, trabalham funcionários que executam as funções de caixa, atendente, auxiliar de limpeza e padeiro (Figura 28).

O senhor João quer que cada cliente receba um cartão com um código na entrada da padaria e que este cartão seja usado para registrar os produtos comprados pelos clientes. Os preços desses produtos deverão ser somados automaticamente assim que o cartão for entregue no caixa, que confirmará o valor total da compra, verificará a forma de pagamento escolhida, receberá o pagamento e, se for o caso, devolverá o troco ao cliente.

O senhor João também deseja controlar dos estoques para que não faltem produtos. Ele tem, portanto, necessidade de informações sobre:

- As vendas, isto é, precisa que sejam armazenados todos os dados de todas as vendas da padaria: quais produtos foram vendidos, em qual quantidade e

Figura 28
O senhor João tem necessidade de informações precisas sobre a sua padaria.



por qual valor, além de qual empregado registrou a venda e qual recebeu o pagamento.
- O estoque, de modo que cada produto vendido seja debitado no saldo.
- A necessidade de reposição de produtos – o modelo deve ter capacidade para gerar a qualquer momento a relação dos itens cujo saldo está abaixo do estoque mínimo desejável para facilitar a identificação daqueles que precisam ser repostos.
- A durabilidade e o uso dos cartões.
- Seus fornecedores – endereços, telefones e o nome do contato na empresa para efetuar a compra.

Uma observação importante: o senhor João já possui um controle fiscal e contábil de toda a movimentação, cujos documentos e registros ele envia semanalmente para seu contador. Fica, assim, para o modelo a ser proposto apenas o controle físico (estoque) e financeiro das transações.

Vamos pensar no problema proposto pelo senhor João, construir um modelo e demonstrá-lo por meio de um diagrama de entidade e relacionamento. Para isso, é preciso dar vários passos.

Passo 1

Listar as entidades candidatas a integrante do modelo.
Quando recebemos uma solicitação nesse formato, isto é, um texto que descreve a situação a ser tratada no modelo, uma prática bastante usada é no próprio texto fazer a identificação das possíveis entidades e relacionamentos, assim como dos principais atributos, grifando os substantivos e verbos essenciais para depois analisá-los a fim de atribuir-lhes os devidos papéis. Vejamos o que podemos selecionar em nosso texto.
No primeiro parágrafo temos os seguintes substantivos: senhor João, pães, produtos, frios, laticínios, lanches, refrigerantes, sorvetes, balas, chicletes, chocolates, cartões telefônicos, produtos, frango assado, balcão, caixa.
No segundo parágrafo encontramos: padaria, funcionários, funções, caixa, atendente, auxiliar de limpeza e padeiro.
No terceiro parágrafo podemos grifar: senhor João, clientes, cartão, código, produtos, padaria, caixa, valor total da compra, forma de pagamento, troco.
No quarto parágrafo identificamos: produto, senhor João, fornecedores.

Passo 2

Analisar e selecionar as entidades que realmente fazem parte do modelo, excluindo as demais.
Vamos avaliar os substantivos apenas uma vez, mesmo se eles aparecerem mais vezes, ou em mais de um parágrafo. Se forem exatamente iguais, será considerada a primeira análise.
Senhor João: é o dono da padaria e pode ser tratado como informação, pois também atende na padaria e trabalha no caixa. Não é entidade.
Pães, frios, laticínios, lanches, refrigerantes, sorvetes, balas, chicletes, chocolates, cartões telefônicos: tipos de produtos vendidos na padaria. São informações relacionadas à entidade produto, mas não são entidades.
Frango assado: é um produto vendido na padaria. Não é entidade.
Balcão: local físico onde ficam expostas as mercadorias. Não é informação.
Caixa: local físico na padaria no qual são registradas e pagas as compras. Não é informação.
Produtos: são os itens que o senhor João vende em sua padaria. Contêm um conjunto de atributos, tais como descrição, saldo e preço de venda. São uma entidade.
Padaria: é o tipo do estabelecimento que o senhor João possui. Tem um conjunto de atributos, como nome, endereço etc. É uma entidade.
Funcionários: são as pessoas que executam algum tipo de serviço necessário ao bom funcionamento da padaria. Possuem uma série de atributos que precisam ser armazenados para facilitar o controle e a consulta de suas informações. São uma entidade.
Funções: referem-se à qualificação do funcionário e ao tipo de serviço que ele exerce na padaria, logo, são atributos de funcionário.
Caixa, atendente, auxiliar de limpeza e padeiro: São os nomes das funções dos funcionários, informações que podem se relacionar com o atributo função.
Clientes: são os agentes de nosso modelo, aqueles que compram os produtos do senhor João. Possuem um conjunto de atributos tais como nome, endereço, telefone. Constituem uma entidade.
Cartão: é o item que representará o cliente na padaria. Demanda o controle de sua durabilidade e de seu uso. É associado ao registro das vendas e possui os atributos código, data de início de uso e data de fim de uso. É uma entidade.
Código: é um atributo da entidade cartão.
Valor total da compra: informação relevante da compra. E, assim, atributo da entidade compra.
Forma de pagamento: informação a opção de pagamento do cliente. É um atributo da entidade compra.
Troco: a diferença entre o valor da compra e o valor dado em dinheiro pelo cliente para o pagamento da compra. Atributo de compra.
Fornecedores: são os responsáveis por fornecer ao senhor João os produtos que ele vende. Possuem um conjunto de informações relevantes, como nome, endereço, telefone para contato. É uma entidade.

Assim, listamos como entidades: produtos, padaria, funcionários, clientes, compra, cartão e fornecedores. Como a boa prática manda nominar as entidades como substantivos no singular, teremos: produto, padaria, funcionário, cliente, cartão, forma de pagamento e fornecedor.

Analisando agora as **entidades** e pensando em sua relevância, temos que:

Chegamos então a uma lista final de quatro entidades:
- **Produto**
- **Funcionário**
- **Cartão**
- **Fornecedor**

- A entidade padaria deve ser retirada de nosso modelo. Como a ideia é criar o modelo apenas para uma padaria, essa pode ficar fora de nosso escopo.

- A entidade cliente também pode ser retirada de nosso modelo, pois, entre as funcionalidades que o senhor João nos solicitou, não consta identificar o que cada cliente comprou. Você já se cadastrou em alguma padaria em que foi fazer compra? Na maioria dos casos, isso não acontece. Por isso, em nosso caso, o cliente será substituído pelo cartão.

Passo 3

Analisar os relacionamentos entre as entidades.
Quanto aos relacionamentos entre as entidades listadas, identificamos:

- **Cartão compra Produto**
- **Funcionário atende (cartão compra Produto)**
- **Fornecedor Fornece Produto.**

Observe que identificamos anteriormente compra como sendo uma entidade e agora listamos-a como sendo um relacionamento. Por quê? Por se tratar de um relacionamento com campos que quando da criação do projeto físico torna-se uma tabela, assim como as entidades.

Passo 4

Definir a cardinalidade dos relacionamentos
Para os relacionamentos definidos, há as seguintes cardinalidades:

Cartão compra Produto
(Senhor Antonio entra na padaria para comprar um litro de leite. Recebe um cartão, vai ao balcão, pega o leite e 100 g de pão de queijo.)
Um cartão (o cartão que senhor Antonio pegou na entrada da padaria) compra vários produtos (um litro de leite e 100 g de pão de queijo) e um produto (leite) pode ser comprado por vários cartões (os da senhora Maria, do senhor Antonio, do Miro). Logo, a cardinalidade desse relacionamento é N para N, sendo necessário que se definam os atributos do relacionamento compra por se tratar de um relacionamento com campos.

Figura 29
Cartão compra produto.

Funcionário atende (cartão compra Produto)
Imagine a cena: a senhora Maria comprou dez pãezinhos e foi atendida pelo funcionário Laércio. Depois dela, Laércio atendeu o senhor Joaquim, Mariana, Pedro.

Figura 30
Funcionário atende.

Portanto, Laércio irá registrar dez pãezinhos no cartão da senhora Maria, depois registrará as compras do senhor Joaquim, Mariana, Pedro.

Logo, um funcionário (Laércio) atende vários (cartão compra produto – o cartão da senhora Maria x dez pãezinhos), mas um (cartão compra produto – o cartão da senhora Maria x dez pãezinhos) só é atendido (a cada vez que ela compra um produto na padaria) por um funcionário (Laércio), logo a cardinalidade deste relacionamento é N para 1.

Fornecedor fornece produto
O senhor Cardoso é dono de um atacado que vende bolachas e chocalates com o melhor preço da cidade. Toda vez que o dono da padaria, o senhor João, precisa comprar bolachas ou chocolates, ele liga para o senhor Cardoso e faz o pedido. No máximo até o dia seguinte o caminhão do senhor Cardoso para em frente à padaria e entrega as mercadorias solicitadas.

Logo, um fornecedor (senhor Cardoso) fornece vários produtos (bolachas e chocolates), e um produto (chocolate ao leite) pode ser vendido por vários fornecedores (senhor Cardoso, Maria Doceria, Bazar dos Amigos). Assim, a cardinalidade deste relacionamento é N para N.

É necessário que se definam os atributos do relacionamento fornece por se tratar de um relacionamento com campos.

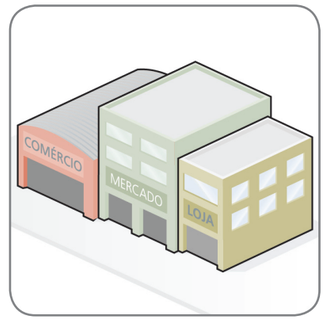


Figura 31
Fornecedor.

Passo 5

Definir as restrições de integridade dos relacionamentos.
Ao identificar tais restrições de integridade, vamos também definir o valor mínimo e máximo de cada cardinalidade.

Cartão compra produto
As restrições de integridade são: um cartão pode comprar ao menos 0 (quando o cartão acabou de ser posto em uso) e no máximo N produtos (todos os produtos dos clientes que pegarem aquele cartão). Já um produto pode ser comprado por no mínimo 0 (o produto acabou de ser lançado e acabou de ser colocado na prateleira) e no máximo N cartões (todos os clientes da padaria).

Logo, as restrições de integridade são (0,N) e (0,N).

Funcionário atende (cartão compra Produto)
Um funcionário pode atender no mínimo 0 – o funcionário acaba de começar a trabalhar na padaria do senhor João – e no máximo N, Senhor Virgílio trabalha há quinze anos na padaria do senhor João. A senhora Maria acaba de chegar à padaria e Laércio, que está no balcão, vai atendê-la. Assim, o cliente pode ser atendido por no mínimo 1 e no máximo 1 funcionário. Logo, as restrições de integridade são (0,N) e (1,1).

Fornecedor fornece Produto
Um fornecedor fornece no mínimo 0 (Mário vende doces mas seus preços são sempre mais caros) e no máximo N produtos (senhor Cardoso). Já um produto (chocolate ao leite) é fornecido por no mínimo 1 e no máximo N fornecedores (senhor Cardoso, Maria Doceria, Bazar dos Amigos). Logo, as restrições de integridade são: (0,N) e (1,N).

Passo 6

Definir os atributos das entidades e relacionamentos com campos e as chaves primária e estrangeira (se houver).
Para definir esses atributos temos de lembrar sempre do conceito de abstração, tentando selecionar apenas os aspectos relevantes para o escopo do problema em foco.

Entidades:
- **Cartao(codigo,data_inicio_uso,data_fim_uso):** o atributo código foi definido como chave primária, pois precisamos ter a certeza de que não existem dois cartões com o mesmo número e compartilharemos a responsabilidade dessa conferência com o sistema gerenciador de banco de dados.
- **Produto(codigo,nome,preco_venda,saldo,estoque_minimo):** o atributo código é chave primária, pois assim fica mais fácil fazer o controle para impedir que o valor deste atributo (codigo) se repita.
- **Funcionario(codigo,nome,funcao):** com o atributo codigo como chave primária.
- **Fornecedor(codigo,nome,rua,complemento,bairro,cidade,estado,cep, contato,telefone,celular):** o atributo codigo é chave primária.

Relacionamentos com campos.
Devemos nos lembrar de que um relacionamento com campos deve conter pelo menos as chaves primárias das entidades envolvidas. Eles podem conter outros atributos.
- **Compra(numero,data,forma_pagamento,codigo_produto,codigo_cartao, quantidade,valor_unitario,valor_total_compra).** Aqui as chaves primárias são os atributos numero, codigo_cartao e codigo_produto e as chaves estrangeiras os atributos codigo_cartao e codigo_produto.
- **Fornece(numero_nota,data,codigo_fornecedor,codigo_produto, quantidade, valor_unitario).** Nesse caso, as chaves primárias são os atributos numero_nota, codigo_fornecedor e codigo_produto e as chaves estrangeiras codigo_fornecedor e codigo_produto.

Passo 7 Desenhar o diagrama de entidade e relacionamento

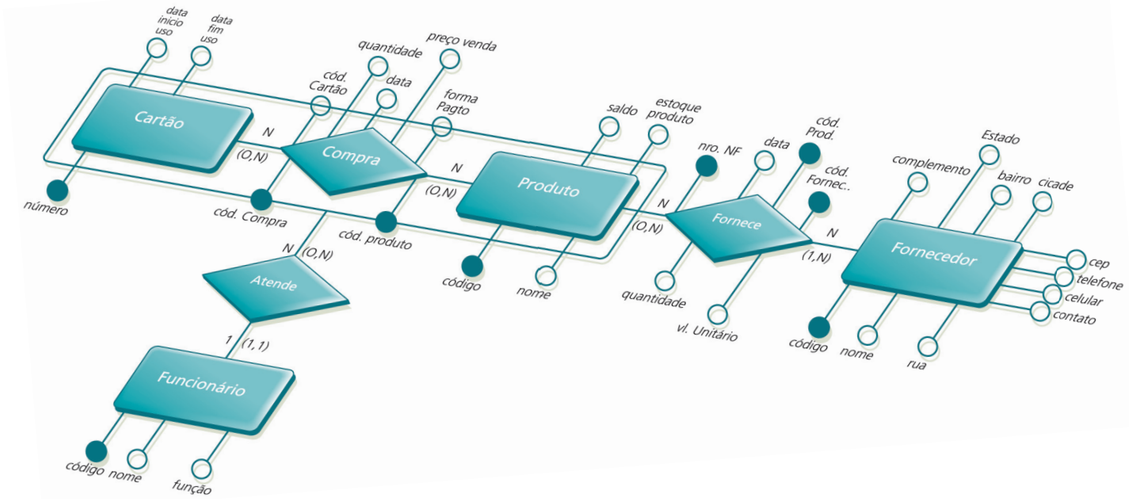


Figura 32
Diagrama de entidade e relacionamento da padaria do senhor João.

2.2. Normalização

Normalização é um processo utilizado para acertar possíveis problemas estruturais das entidades e relacionamentos com campos criados – também chamados de anomalias – em um modelo de entidade e relacionamento. Consiste na análise dos atributos das entidades e relacionamentos com campos, sob o ponto de vista das regras chamadas formas normais, que descrevem, com base na teoria de conjuntos, na álgebra e no cálculo relacional, o que devemos ou não fazer nas estruturas das entidades e relacionamentos de nosso modelo, baseados em conceitos matemáticos.

Essa análise pode demonstrar a necessidade de alterarmos a estrutura de nossas entidades e relacionamentos com campos, dividindo ou agrupando seus atributos para aprimorar o processo de recuperação das informações (performance) e seu armazenamento, de modo a evitar perda, redundância e distorção da informação.

Sempre que formos obrigados pela aplicação das formas normais em nosso modelo a dividir entidades, temos que garantir que a divisão poderá ser revertida, isto é, que, mesmo particionada em duas ou mais entidades, uma entidade poderá voltar à sua formação original, por meio de operações de conjuntos.

Mas, antes de tratar das regras de normalização propriamente ditas, é necessário entender alguns conceitos da álgebra relacional, que serviram para definir se nossas entidades e relacionamentos estão ou não em uma forma normal.

Dependência funcional
Seja R uma relação e X e Y atributos de R. X e Y podem ser atributos simples ou compostos.

X → Y (o atributo X determina funcionalmente o atributo Y) sempre que duas tuplas quaisquer de R tiverem o mesmo valor para X, elas possuem também o mesmo valor para Y.

Exemplo:
Tendo a entidade funcionario os atributos codigo, nome, cidade e DDD, e sabendo que o codigo é a chave primária da entidade funcionario, se analisarmos esses atributos sob a óptica da dependência funcional, teremos:

codigo → nome
codigo → cidade
cidade → DDD

Logo, podemos dizer que os atributos nome e cidade dependem do atributo codigo. Já o atributo DDD depende do atributo cidade.

Definida a dependência funcional, podemos passar agora para a definição das formas normais.

Primeira Forma Normal (1NF)

Uma entidade está em Primeira Forma Normal, se e somente todos os seus atributos são atômicos, isto é, se contém um valor único (atômico) e não contém atributos multivalorados.

Exemplo:
Dada a entidade funcionario, definida com os atributos abaixo:

Funcionario(codigo,nome,data_admissao,data_demissao,habilidades)

Vemos que a entidade funcionario possui o campo multivalorado habilidades, o que não é permitido pela Primeira Forma Normal. Devemos então dividir a tabela funcionario de forma que o campo habilidades se torne uma nova entidade.

Então teremos:

Funcionario(codigo,nome,data_admissao,data_demissao)
Possui(cod_funcionario,cod_habilidade)
Habilidade(codigo,descricao)

Exemplo:

Fornecedor(codigo,nome,endereco,telefones)

Vemos que a entidade fornecedor tem como atributo composto endereco e como atributo multivalorado telefones.

Em relação ao atributo composto endereco, sabemos que o mesmo é composto, pois nele pressupõe-se incluir as informações de rua, complemento, bairro, cidade, estado e CEP. Ou substituímos o atributo endereco por seus atributos componentes (rua, complemento, bairro, cidade, estado e CEP) ou criamos uma outra entidade com o nome do atributo composto (endereco), tendo como atributos dessa nova entidade rua, complemento, bairro cidade, estado e CEP.

Já o campo telefones, por estar no plural, indica que nele poderá ser armazenado mais de um número. Pela regra, esse atributo precisa ser separado em outra entidade, que pode ser chamada de telefone e que conterá os diversos números de telefone do fornecedor.

Assim, temos:

Fornecedor(codigo,nome,rua,complemento,bairro,cidade,estado,cep)

Tendo como chave primária o atributo codigo.

Telefone(cod_fornecedor,nro telefone)

Tendo como chave primária os atributos cod_fornecedor e nro_telefone, já que isso garante que não será cadastrado o mesmo telefone para o forncedor e como chave estrangeira o atributo cod_fornecedor.

Fornecedor(codigo,nome)

Tendo como chave primária o atributo código.

Endereco(cod_fornecedor,rua,complemento,bairro,cidade,estado,cep).

Tendo como chave primária o atributo cod_fornecedor e como chave estrangeira o atributo cod_fornecedor.

Segunda Forma Normal (2NF)

Uma entidade encontra-se em Segunda Forma Normal se e somente estiver em Primeira Forma Normal e não tiver atributos com dependências parciais. No caso de uma chave primária composta, isto é, que possui mais de um atributo em sua composição, é denominada dependência parcial a dependência de um atributo não chave a apenas uma parte da chave primária.

Toda entidade que não possuir chave primária composta, isto é, com mais de um atributo, está em 2ª Forma Normal.

Tomemos como exemplo a tabela compra, descrita abaixo:

Compra(nro_nf,cod_fornecedor,cod_produto,data,nome produto, quantidade,valor_unitario,valor_total_nota)

Tendo como chave primária os atributos: nro_nf, cod_fornecedor, cod_produto
Se analisarmos a dependência funcional teremos:

nro_nf,cod_fornecedor → data
nro_nf,cod_produto → quantidade
nro_nf,cod_produto → valor_unitario
nro_nf,cod_fornecedor → valor_total_nota
cod_produto → nome_produto

Vemos agora que nem todos os atributos dependem da chave primária. O que não é permitido pela Segunda Forma Normal. Para que essa entidade fique em 2NF, teremos de desmembrá-la.

Ela ficará assim:

Compra(nro_nf,cod_fornecedor,data,valor_total_nota)

Tendo como chave primária os atributos nro_nf cod_fornecedor e como chave estrangeira o atributo cod_fornecedor.

Item_compra(nro_nf,cod_produto,quantidade,vl_unitario)

Tendo como chave primária os atributos nro_nf e cod_produto e como chaves estrangeiras os atributos nro_nf e cod_produto.

Produto(codigo,nome)

Tendo como chave primária o atributo codigo.

Terceira Forma Normal (3NF)

Uma entidade está em Terceira Forma Normal se e somente estiver em Primeira e em Segunda Forma Normal e todos os atributos não chave dependerem funcionalmente da chave primária.

Exemplo:

Pedido(nro_pedido,data,cod_cliente,nome_cliente,email_cliente,valor_total_pedido)

Vamos verificar a dependência funcional dos atributos:

nro_pedido → data
nro_pedido → cod_cliente
nro_pedido → valor_total_pedido
cod_cliente → nome_cliente
cod_cliente → email_cliente

Verificamos que os atributos nome_cliente e email_cliente não são dependentes da chave primária e sim do atributo cod_cliente. Será necessário então desmembrar a entidade pedido.

Pedido(nro_pedido,data,cod_cliente,valor_total_pedido)

Que terá como chave primária o atributo nro_pedido.

Cliente(cod_cliente,nome_cliente,email_cliente)

Que terá como chave primária o atributo cod_cliente.

Forma Normal Boyce-Codd (BCNF)

Uma entidade está em BCNF se e somente estiver em Terceira Forma Normal e todos os atributos não chave dependerem apenas da chave primária.

Exemplo:

Cliente(cod_cliente,nome_cliente,email_cliente)

Que terá como chave primária o atributo cod_cliente.

cod_cliente → nome_cliente
cod_cliente → email_cliente

Todos os atributos não chave dependem funcionalmente apenas da chave primária. Logo, está em BCNF.

Vamos agora, como mais um exemplo de aplicação das regras de normalização, aplicar a normalização nas entidades do modelo da padaria do senhor João.

Ao final da montagem de nosso modelo ficamos com as seguintes entidades:

Cartao(codigo,data_inicio_uso,data_fim_uso). A chave primária foi definida como sendo o atributo codigo.

Produto(codigo,nome,preco_venda,saldo,estoque_minimo), tendo o atributo codigo como chave primária.

Funcionario(codigo,nome,funcao), tendo o atributo codigo como chave primária.

Fornecedor(codigo,nome,rua,complemento,bairro,cidade,estado,cep,contato,telefone,celular), tendo o atributo codigo como chave primária.

Compra(numero,data,forma_pagto,codigo_produto,codigo_cartao, quantidade,valor_unitario,valor_total_compra), tendo como chave primária os atributos numero, codigo_cartao e codigo_produto e como chaves estrangeiras os atributos codigo_cartao e codigo_produto.

Fornece(numero_nota,data,codigo_fornecedor,codigo_produto, quantidade, valor_unitario), tendo como chave primária os atributos numero_nota, codigo_fornecedor e codigo_produto e como chaves estrangeiras codigo_fornecedor e codigo_produto.

Cartao(codigo,data_inicio_uso,data_fim_uso) – a chave primária foi definida sendo o atributo codigo.

Agora vamos analisar cada uma das entidades.

OBSERVAÇÕES

• Sempre que for preciso desmembrar uma entidade, isso deve ser feito de maneira que seja possível retornar à forma anterior, evitando, com isso, a perda de informações.

• Sempre que se fizer o desmembramento de uma entidade, as tabelas resultantes devem ser submetidas novamente às formas normais para se ter certeza de que estão corretamente normalizadas.

Entidade cartão

Cartao(codigo,data_inicio_uso,data_fim_uso)

O atributo código foi definido como chave primária.
Está em Primeira Forma Normal, pois todos os seus atributos são atômicos.
A entidade cartao está em Segunda Forma Normal, pois sua chave primária não é composta.

Vamos verificar a dependência funcional da entidade cartao:

codigo→data_inicio_uso
codigo→data_fim_uso

Logo, a entidade cartao está em Terceira Forma Normal, pois todos os atributos são dependentes funcionalmente da chave primária.

A entidade cartao está na Forma Normal Boyce-Codd, pois todos os seus atributos não chave dependem unicamente da chave primária.

Cartao(codigo,data_inicio_uso,data_fim_uso)

Tendo o atributo codigo como chave primária.

Entidade Produto

Produto(codigo,nome,preco_venda,saldo,estoque_minimo)

Tendo o campo codigo como chave primária.

Está em Primeira Forma Normal, pois ela não possui atributos multivalorados e atributos compostos.

A entidade produto está em Segunda Forma Normal, pois sua chave primária não é composta. Analisando sua dependência funcional, temos:

codigo →nome
codigo →preco_venda
codigo →saldo
codigo →estoque_minimo

A entidade produto está em Terceira Forma Normal, pois todos os atributos não chave dependem funcionalmente da chave primária. A entidade produto está na Forma Normal Boyce-Codd, pois todos os atributos não chave são dependentes apenas da chave primária.

Entidade funcionario

Funcionario(codigo,nome,funcao)

Tendo o atributo codigo como chave primária.

Está em Primeira Forma Normal, pois não possui atributos multivalorados nem atributos calculados.

A entidade funcionario está em Segunda Forma Normal, pois não possui chave primária composta.

Verifiquemos a dependência funcional da entidade funcionario:

codigo →nome
codigo →funcao

A entidade funcionario está em Terceira Forma Normal, pois todos os seus atributos não chave são dependentes da chave primária.

A entidade funcionario está na Forma Normal Boyce-Codd, pois todos os atributos não chave dependem apenas da chave primária.

Funcionario(codigo,nome,funcao)

Tendo o campo codigo como chave primária

Entidade Fornecedor

Fornecedor(codigo,nome,rua,complemento,bairro,cidade,estado,cep,contato,telefone,celular)

Tendo o atributo codigo como chave primária.

Está em Primeira Forma Normal, pois não possui atributos multivalorados nem atributos calculados.

A entidade fornecedor está em Segunda Forma Normal, pois sua chave primária é simples.

Análise da dependência funcional da entidade fornecedor.

codigo → nome
codigo → rua
codigo → complemento
codigo → bairro
codigo → cidade
codigo → estado
codigo → cep
codigo → contato
codigo →telefone
codigo →celular

- A entidade fornecedor está em Terceira Forma Normal, pois todos os atributos não chave dependem funcionalmente da chave primária.
- A entidade fornecedor está na Forma Normal Boyce-Cood, pois todos os seus atributos não chave dependem apenas da chave primária.

Entidade compra

Compra(numero,data,forma_Pagto,codigo_produto,codigo_cartao,quantidade,valor_unitario,valor_total_compra)

Tendo como chave primária os atributos numero, codigo_cartao e codigo_produto e como chaves estrangeiras os atributos codigo_cartao e codigo_produto.

Está em Primeira Forma Normal, pois todos os seus atributos são atômicos.

Verificando a dependência funcional da entidade compra, temos:

numero,codigo_cartao,cod_produto → data
numero,codigo_cartao,cod_produto → forma_Pagto
numero,cod_produto → quantidade
numero,cod_produto → valor_unitario

Nem todos os atributos dependem da chave primária, então, para deixar a entidade compra em Segunda Forma Normal, devemos desmembrá-la (compra e ItemCompra), assim:

Compra (numero, cod_cartao,data,valor_total_compra,forma_pagto)

E com a chave primária contendo o atributo numero e a chave estrangeira cod_cartao.

ItemCompra(numero,cod_produto,quantidade,valor_unitario)

Com chave primária contendo os atributos numero e cod_produto e como chaves estrangeiras os atributos numero e cod_produto.

- A entidade Compra encontra-se em Segunda Forma Normal.
- A entidade ItemCompra encontra-se em Segunda Forma Normal.
- A entidade Compra encontra-se em Terceira Forma Normal, pois como vimos na verificação da dependência funcional, todos os atributos não chave dependem da chave.
- A entidade Compra está na Forma Normal Boyce-Codd, pois todos os atributos não chave dependem unicamente da chave primária.
- A entidade ItemCompra está na forma normal Boyce-Codd, pois todos os atributos não chave dependem unicamente da chave primária.

Entidade fornece

Fornece (numero_nota, data, codigo_fornecedor,codigo_produto, quantidade, valor_unitario)

Tendo como chave primária os atributos numero_nota, codigo_fornecedor e codigo_produto e como chaves estrangeiras codigo_fornecedor e codigo_produto.

A entidade Fornece está em Primeira Forma Normal, pois todos os seus atributos são atômicos. Vale verificar a dependência funcional dessa entidade.

numero_nota,cod_fornecedor,cod_produto → data
numero_nota,cod_fornecedor,cod_produto → valor_total
numero_nota,cod_produto → quantidade
numero_nota,cod_produto → valor_unitario

Nem todos os atributos não chave dependem completamente da chave, logo é preciso desmembrar seus atributos:

Fornece(numero_nota,cod_fornecedor, data,valor_total)

Ficando como chave primária os atributos numero_nota e cod_fornecedor e como chave estrangeira o atributo cod_fornecedor.

ItemFornece(numero_nota,cod_produto,quantidade,valor_unitario)

Tendo como chave primária os atributos numero_nota e cod_produto e como chaves estrangeiras os atributos numero_nota e cod_produto.

- A entidade Fornece está em Segunda Forma Normal, pois todos os atributos não chave dependem completamente da chave.
- A entidade ItemFornece está em Segunda Forma Normal, pois todos os atributos não chave dependem completamente da chave.
- A entidade Fornece está em Terceira Forma Normal, pois todos os seu atributos não chave dependem da chave primária
- A entidade ItemFornece está em Terceira Forma Normal, pois todos os seus atributos não chave dependem da chave primária.
- A entidade Fornece está na Forma Normal Boyce-Codd, pois todos os atributos não chave dependem unicamente da chave primária.
- A entidade ItemFornece está na Forma Normal Boyce-Codd, pois todos os atributos não chave dependem unicamente da chave primária.

Há outros passos a serem seguidos até o nosso modelo ser implementado, mas, só para entendermos o que as entidades representam, vejamos como ficarão as entidades da padaria do senhor João depois de implementadas. Em nossa representação tabela *Compra*, a primeira linha é o nome da tabela (entidade), a segunda linha os nomes dos campos (atributos) e a partir da terceira linha são as informações armazenadas que chamamos de registros ou tuplas. Veja como os relacionamentos permitem entender perfeitamente as informações gravadas nas tabelas.

COMPRA

numero	cod_cartao	data	forma_pagto	valor_total compra
132	3	21/05/2009	dinheiro	10,60
133	2	21/05/2009	dinheiro	13,60
134	6	23/05/2009	cartao	52,20

Vamos analisar, por exemplo, a tabela ItemCompra. Veja que a soma dos itens da compra de código 132 (campos quantidade *valor_unitario) é o mesmo valor gravado no campo valor_total_compra da tabela compra. Isto é integridade de dados. Aproveite para olhar detalhadamente para as demais tabelas, para entender como os atributos se transformam em campos e como esses campos se relacionam, permitindo acesso rápido e eficiente às informações. Olhe agora para as tabelas Fornecedor, Fornece e ItemFornece. Só de olhar, já sabemos que, no dia 21/05/2009, o senhor João comprou do senhor Pedro Parente 120 litros de leite B e dois achocolatados em pó.

ITEMCOMPRA

numero_nota	cod_produto	quantidade	valor_unitario
132	2	2	3,20
132	3	1	4,20

Como sabemos disso? Veja o caminho que fazemos saindo de Fornece indo para ItemFornece; veja a implementação do relacionamento, pois os campos numero_nota das duas tabelas tem o valor 1 e o código do produto de ItemFornece equivale aos produtos leite B e achocolatado em pó. Vemos agora na prática o que definimos na teoria. Preste atenção nas outras tabelas e veja se você acha mais informações interessantes. Olhe também a importância dos atributos chave primária e chave estrangeira em nossa implementação. Ainda há muito o que aprender para que os modelos se transformem em sistemas de qualidade, mas agora já se sabe como os modelos se transformam em banco de dados de aplicações.

FORNECEDOR

Codigo	Nome	Rua	Complemento	Bairro	Cidade	Estado	CEP	Contato	Telefone	Celular
1	Cardoso Alimentos	Rua 2 nº 32		Jd. Boa Esperança	Campinas	SP	12123-123	Sr. Cardoso	99-3234-0000	99-9999-0000
2	Maria Doceria	Rua 34 nº 123	Atrás da Igreja	Centro	Hortolândia	SP	12123-123	D. Maria		
3	Pedro Parede	Rua 4 nº 120		Jd. Cachoeira	Caldas	MG	04321-789	Sr. Pedro	99-9999-8976	87-9999-0000

FORNECE

numero_nota	cod_fornecedor	data	valor_total
1	3	21/05/2009	123,12
2	2	21/05/2009	34,20
4	6	23/05/2009	48,90

ITEMFORNECE

numero_nota	cod_produto	quantidade	valor_unitario
1	2	120	1,00
1	3	2	1,56

PRODUTO

Codigo	Nome	Preço_Venda	Saldo	estoque_Minimo
2	Leite B.	1,89	32	4
3	Achocolatado em pó	3,45	13	2
15	Leite condensado	2,11	54	12

FUNCIONARIO

Codigo	Nome	Função
1	Sr. João	Dono
2	Laercio da Silva	Padeiro
3	Maria padilha	Atendente

2.3. Fases de um projeto utilizando o modelo ER

No caso fictício da padaria do senhor João, descobrimos, por meio de um relato, como a padaria funcionava e quais eram as expectativas de seu dono em relação a como passaria a operar e ao tipo de informação que o novo sistema lhe permitiria obter. Na vida real, dificilmente acontece aquela sequência de ações idealizadas para compor o exemplo. Geralmente, o usuário (cliente) não sabe muito bem do que precisa, quando decide informatizar algum processo do seu negócio. Nós é que devemos nos aproximar dele para reunir elementos que permitam desenvolver a solução que o atenda da melhor forma. É preciso também oferecê-la de acordo com o escopo esperado, isto é, construída dentro do prazo solicitado e com o orçamento disponível. Sim, porque para viabilizar um projeto essa dualidade é fundamental: a correta conjunção de tempo combinado e cujo valor esteja dentro do investimento que o cliente está disposto a fazer.

Não é, portanto, uma tarefa fácil. Corre-se o risco da perda de foco e de ritmo de trabalho, se não forem usadas técnicas que sinalizem corretamente e facilitem nosso caminho. Felizmente, há um roteiro para a criação de soluções informatizadas que utilizam o Modelo de Entidade e Relacionamento, um guia que pode nos conduzir a um bom resultado final, ou seja, o projeto pronto e instalado na empresa do cliente (figura 33).

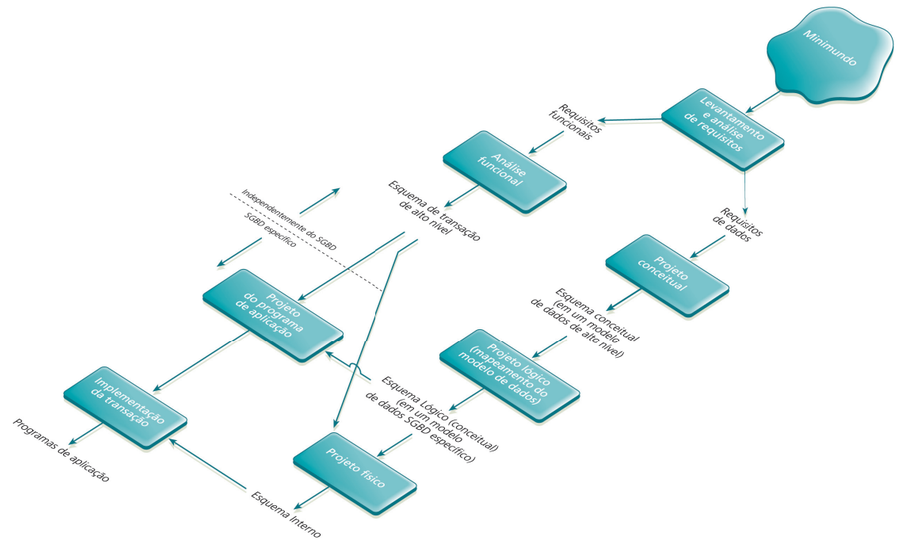


Figura 33
Roteiro para criar uma solução informatizada.

O roteiro oferece uma série de indicações a seguir até que se alcance o produto final – o software implementado. Há, porém, algumas hipóteses que podem alterar os rumos do projeto. Durante seu desenvolvimento, podemos concluir, por exemplo, que a solução inicialmente imaginada não é econômica ou tecnicamente viável; ficará cara demais, sem proporcionar o resultado esperado, no prazo estabelecido. Pode-se até concluir que não há necessidade de uma solução informatizada para o problema proposto, sugerindo que seja resolvido apenas por meio de uma simples mudança de procedimento (inclusão e/ou alteração de ações dos usuários no processo).

Vale, então, conhecer e analisar em detalhes cada um dos passos sugeridos no roteiro, para então aplicá-los ao nosso projeto para a padaria do senhor João. Verificaremos, assim, quais pontos devem ser levados em conta em cada passo e qual será o produto final para cada fase proposta. Comecemos pelo minimundo.

2.3.1. Minimundo

O minimundo – geralmente o ponto inicial do trabalho – é a parte do mundo real que é o foco do projeto ou de nossa análise. No nosso caso, é a padaria do senhor João. Usaremos as técnicas descritas no capítulo 1, para conhecer melhor o funcionamento da empresa, em especial o foco do problema, de acordo com os pontos levantados pelo senhor João: a dinâmica do atendimento ao público, o processo de compra e venda de mercadorias e a necessidade de reposição de estoques no prazo desejável.

2.3.2. Levantamento de requisitos

Por meio das técnicas que estudamos no capítulo anterior, vamos levantar os requisitos para a solução do problema proposto. Avaliaremos, portanto, o tamanho do problema, o que se espera da solução, quem são os usuários chave, quais processos estão envolvidos e o que cada um desses processos oferece de informação relevante à solução que estamos buscando.

Quem pode nos dar essas respostas são os usuários e seus procedimentos. Lembre-se de que devemos escolher uma ou mais técnicas descritas para conhecer bem o problema.

DICA

Desde o início, as informações levantadas no cliente devem ser registradas para que se tornem base de nosso entendimento do problema e referência para consulta posteriores.

De início, podemos recorrer aos métodos de entrevista para obter informações do senhor João e seus funcionários com o objetivo de compreender o funcionamento da padaria. Também podemos observar os funcionários em seu dia-a-dia para verificar a dinâmica do trabalho, suas rotinas, particularidades e informações geradas nas diferentes operações. Para cada entrevista, procedimento ou etapa é necessário produzir uma documentação, isto é, anotar de forma clara e isenta as informações obtidas e suas fontes (quem nos deu tal informação, como chegamos a determinada conclusão). Este procedimento nos permitirá construir uma base de apoio que poderemos consultar sempre que surgir alguma dúvida em relação ao processo ou à solução imaginada. Cada passo também pode nos apontar novas informações a analisar ou procedimentos a seguir, além de meios de sanar dúvidas que eventualmente surjam no caminho e indicações sobre quem pode nos ajudar a dirimi-las. É fundamental, ainda, cultivar a melhor relação possível com as pessoas envolvidas no processo, deixando clara sua importância para o trabalho, mesmo depois que tivermos encerrado a fase de levantamento de requisitos.

2.3.3. Análise de requisitos

Entendida a dinâmica de funcionamento de nosso minimundo e obtidas as informações relevantes ao foco da solução imaginada, é hora de analisar essas informações, separá-las e classificá-las de forma podermos continuar a desenvolvê-la, verificando inclusive se a melhor opção é mesmo informatizar, e, em caso positivo, se haverá mesmo condições de satisfazer as necessidades do usuário no prazo previsto. O primeiro passo, agora, é separar os requisitos levantados e classificá-los em requisitos de dados e requisitos funcionais. Mas, antes de tudo, resta esclarecer bem quais são esses requisitos.

DICA

Lembre-se: podem surgir dúvidas em qualquer fase do desenvolvimento do projeto e você precisará de mais informações e opiniões do usuário até o fim do processo.

2.3.4. Requisitos de dados

Trata-se, aqui, de toda e qualquer informação relevante para a solução em análise, tanto as geradas quanto as consultadas com o objetivo de concluir determinada tarefa. Por exemplo, a quantidade e o valor do produto comprado, ou seja, o montante do pagamento, são requisitos de dados que devem ser classificados para que se possa construir um modelo de dados.

2.3.5. Requisitos funcionais

São aqueles que descrevem funcionalidades e serviços do sistema. Tal definição dependerá do tipo do software, dos resultados esperados e do tipo do sistema em que o software será aplicado. Exemplos: o sistema deve oferecer diversas maneiras de visualizar os dados, de acordo com o perfil do usuário; os relatórios

Figura 34
As várias operações do negócio padaria.

têm de ficar disponíveis para impressão, de acordo com o nível hierárquico de cada um deles. Para saber qual é esse nível, é necessário que ele se identifique no sistema, digamos, por meio de uma rotina de login, em que ele se apresente via senha, geralmente criada por ele mesmo.

Podemos considerar requisitos funcionais também a manutenção, isto é, inclusão/alteração/exclusão e consulta de todas as entidades identificadas na solução.

Observe que narramos a situação atual da padaria do senhor João, isto é, sem a solução informatizada, e a submetemos às regras de requisitos que definimos no capítulo anterior (necessário, não-ambíguo, verificável, conciso, alcançável, completo, consistente, ordenável e aceito). Agora, avaliemos as operações envolvidas nas vendas (figura 34).

Situação
A senhora Maria vai até o balcão de pães e solicita dez pãezinhos ao funcionário Laércio, que está atendendo naquele momento.
Laércio pega o saco de papel adequado a tal quantidade, vai até a cesta de pães e com o pegador coloca no saco os dez pãezinhos. Em seguida fecha o saco, coloca-o sobre a balança e digita, no equipamento, o preço do quilo do pão. Toma então um pedaço de papel que está sobre o balcão no qual anota o peso dos pães e o valor apresentado na balança, além da palavra pão, e entrega o pacote e o pedaço de papel à senhora Maria, perguntando-lhe se vai querer mais alguma coisa.

Como separar requisitos de dados de requisitos funcionais nesta situação ?
Primeiro, vamos tentar simplificar o problema, selecionando apenas as informações relevantes. Para isso, vamos reconstruir quadro a quadro a situação, formulando algumas perguntas:

1 - A senhora Maria vai até o balcão de pães e solicita dez pãezinhos ao funcionário Laércio, que está no balcão neste momento.
Quais são os requisitos importantes para nossa solução nesta frase?

A senhora Maria se deslocar até o balcão de pães não é um requisito relevante, pois não nos interessa, no momento, como ela chegou ao balcão e sim o que fez ao chegar lá.

2 - Solicita dez pãezinhos ao funcionário Laércio que está atendendo naquele momento.
Ou seja, a senhora Maria (a cliente) solicita dez pãezinhos (dez unidades do produto pãozinho) para o funcionário Laércio que está no balcão.

Concluímos, aqui, que o cliente solicita uma certa quantidade de um determinado produto a certo funcionário.

Laércio ser o funcionário que está no balcão no momento, e a senhora Maria a cliente, são informações que nos demostram apenas que há um funcionário e um cliente envolvidos na ação.

Continuando nossa análise:

3 - Laércio pega o saco de papel adequado a tal quantidade, vai até a cesta de pães e com o pegador coloca no saco os dez pãezinhos. Em seguida, fecha o saco, coloca-o sobre a balança e digita no equipamento o preço do quilo do pão.

Temos de importante aqui que o funcionário pesa o produto, digitando o preço do quilo na balança.

4 - Toma, então, um pedaço de papel que está sobre o balcão no qual anota o peso dos pães e o valor apresentado na balança, além da palavra pão; entrega o pacote e o pedaço de papel à senhora Maria, perguntando-lhe se vai querer mais alguma coisa.

Aqui temos que:
funcionário anota a quantidade, o tipo de produto
e seu preço em um pedaço de papel e o entrega ao cliente.

Em resumo:

- O cliente solicita uma quantidade de um certo produto ao funcionário.
- O funcionário pesa o produto, digitando o preço do quilo na balança.
- O funcionário anota a quantidade, o produto e seu preço em um pedaço de papel e o entrega ao cliente.

Antes de separarmos os requisitos funcionais dos requisitos de dados, precisamos pensar se as três ações descritas no resumo fazem parte do escopo de nosso projeto. Vejamos.

O cliente pedir o produto e o empregado separá-lo não são fatos relevantes na situação, pois são ações que não serão alteradas: o cliente continuará a pedir os produtos aos funcionários e os procedimentos a seguir continuarão sendo os mesmos.

O que importa, então, é que o funcionário anota a quantidade, o produto e seu preço em um pedaço de papel e o entrega ao cliente.

Aí temos uma ação que será modificada por nossa solução, pois sabemos que o senhor João quer que o cliente entregue um cartão ao funcionário, que nele registrará as compras e o devolverá ao cliente.

Os requisitos de dados contidos nesta situação são: funcionário, quantidade de produto, valor total de produto, nome do produto e do cliente. Já o requisito funcional é: anota, pois o funcionário deverá anotar os itens comprados pelo cliente.

Requisitos colhidos até agora.
Ao terminar a análise dessa parte do problema, teremos:

- Requisitos de dados: funcionário, quantidade comprada, valor total do produto e descrição do produto.
- Requisitos funcionais: funcionário anota itens solicitados pelo cliente, isto é, o funcionário deverá ter um "lugar" no sistema para anotar as compras do cliente.

2.3.6. Projeto conceitual

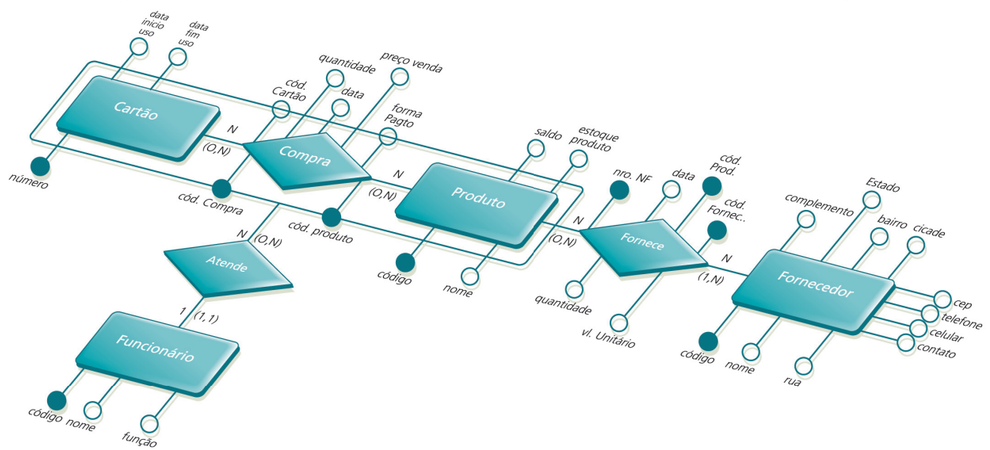
Depois de analisar e classificar todos os requisitos levantados, devemos nos concentrar nos requisitos de dados, agrupando-os em entidades e relacionamentos. Ao pensarmos nas entidades, devemos verificar quais são as informações relevantes em vários aspectos, seguindo os sete passos propostos no tópico Modelo de entidade e relacionamento, estudado no início deste capítulo, para montar o diagrama de entidade e relacionamento, que vai retratar nosso modelo conceitual e classificará as entidades geradas segundo as regras de normalização. Nosso projeto conceitual ficará como sugere a figura 35.

2.3.7. Projeto lógico

Figura 35
A aplicação do diagrama de entidade e relacionamento, na prática.

Com o projeto conceitual montado, precisamos, agora, definir o Sistema Gerenciador de Banco de Dados (SGBD) que empregaremos para implementar nossa solução de software. Existem inúmeras opções de SGBD no mercado, em uma gradação de valor que vai dos gratuitos aos bastante caros. Todos implicam vantagens e desvantagens, que você deverá analisar juntamente com os demais participantes do projeto.

Diagrama de entidade e relacionamento da padaria do sr. João



Depois de escolher o SGBD, devemos verificar quais são os tipos de dados que este aceita, bem como seus tamanhos, pois tais características variam muito. Existem quatro tipos básicos de dados: texto, número, data e hora. Para a padaria do senhor João, usaremos o Banco de dados MySQL. Com essa definição, podemos passar à etapa seguinte: fazer o projeto lógico das entidades e relacionamentos com os campos que usaremos na solução. É preciso definir os atributos de cada entidade, além de tipo, tamanho e obrigatoriedade ou não de cada atributo, e, ainda, avaliar se o atributo é chave primária ou estrangeira e se é único. Resta conhecer o significado das classificações dos atributos (Leia o quadro abaixo).

SIGNIFICADOS E DICAS	
Nome	palavra que indique o atributo no contexto da entidade; deve-se evitar, o uso de abreviações e números.
Tipo	indica a forma como o atributo será armazenado (data, texto, número, etc.).
Tamanho	expressa o número de caracteres que o atributo ocupará na entidade. É preciso cuidado para definir o tamanho do atributo para que não se crie problemas difíceis de resolver no futuro, pois alterar o tamanho de um atributo, depois que o sistema está em produção, implica alterar todas as telas e relatórios nos quais tal atributo aparece. Por exemplo, se criarmos o campo código da tabela cliente com 3 posições, poderão ser cadastrados no máximo 999 clientes, o que representará uma limitação no sistema.
Obrigatório	define se este atributo tem de ser preenchido para que se possa incluir uma informação nesta entidade ou não.
Único	o atributo deve ser definido como único, quando seus valores não puderem ser repetidos.
Chave primária	atributo ou conjunto de atributos que definem um único registro (tupla) em uma entidade.
Chave estrangeira	atributo ou conjunto de atributos que garante o relacionamento entre duas ou mais entidades. Assegura o que chamamos de integridade referencial, isto é, se as tabelas devem ou não possuir uma informação cadastrada para permitir seu uso em outra tabela.
Valor default	indica se o atributo tem um valor padrão de preenchimento.
Regra de validação	demonstra se o atributo possui ou não uma regra de preenchimento. Por exemplo: o atributo sexo só pode receber valores M ou F.

Tabelas

São inúmeras as formas de apresentar as definições alinhadas, e utilizaremos a mais comum, a tabela. Então, por meio de tabelas, vamos criar o modelo lógico das entidades da solução imaginada para a padaria do senhor João. As entidades e relacionamentos com campos passam a ser chamados de tabelas e seus atributos de campos. Observe o exemplo, na tabela *Cartão*.

CARTAO

nome	tipo de dados	tamanho	obrigatorio	unico	chave primaria	chave estrangeira	valor default	regra de validacao
codigo	INTEGER	8	Sim	Sim	Sim	Não	Não	Não
dt_Inicio_uso	DATE	8	Sim	Não	Não	Não	Não	Não
dt_fim_uso	DATE	8	Não	Não	Não	Não	Não	Não

Observação: O campo dt_fim_uso foi definido como não obrigatório, pois nenhum dos cartões, quando de seu cadastro, tem data indicando fim de prazo de validade. Veja as tabelas *Produto*, *Funcionario*, *Fornecedor*, *Fornece*, *ItemFornece* e *Compra*.

PRODUTO

nome	tipo de dados	tamanho	obrigatorio	unico	chave primaria	chave estrangeira	valor default	regra de validacao
codigo	INTEGER	5	Sim	Sim	Sim	Não	Não	Não
nome	VARCHAR	40	Sim	Não	Não	Não	Não	Não
preco_venda	DECIMAL	10,2	Sim	Não	Não	Não	Não	Valor > 0
saldo	DECIMAL	10,2	Sim	Não	Não	Não	0	Valor >=0
estoque_minimo	DECIMAL	10,2	Sim	Não	Não	Não	0	Valor >=0

Observação: O campo saldo foi definido com decimal, porque nem sempre os produtos são vendidos em unidades.

FUNCIONARIO

nome	tipo de dados	tamanho	obrigatorio	unico	chave primaria	chave estrangeira	valor default	regra de validacao
codigo	INTEGER	4	Sim	Sim	Sim	Não	Não	Não
nome	VARCHAR	50	Sim	Não	Não	Não	Não	Não
função	VARCHAR	30	Sim	Não	Não	Não	Não	Não

O melhor SGBD

Definir qual Sistema Gerenciador de Banco de Dados devemos adotar em uma solução informatizada nem sempre é fácil. Precisamos levar em conta seu preço, sua forma de comercialização, seus custos adicionais (valor da manutenção anual, preço por usuário etc.), estrutura de hardware que a solução demandará (rede, stand-alone, internet), grau de segurança, tipos de acesso, formas de gerenciamento de informações. Todos esses fatores têm de ser considerados, para chegar à melhor solução.

FORNECEDOR

nome	tipo de dados	tamanho	obrigatorio	unico	chave primaria	chave estrangeira	valor default	regra de validacao
codigo	INTEGER	4	Sim	Sim	Sim	Não	Não	Não
nome	VARCHAR	50	Sim	Não	Não	Não	Não	Não
rua	VARCHAR	50	Sim	Não	Não	Não	Não	Não
complemento	VARCHAR	50	Não	Não	Não	Não	Não	Não
bairro	VARCHAR	40	Não	Não	Não	Não	Não	Não
cidade	VARCHAR	40	Sim	Não	Não	Não	Não	Não
estado	VARCHAR	2	Sim	Não	Não	Não	Não	Não
cep	VARCHAR	8	Sim	Não	Não	Não	Não	Não
telefone	VARCHAR	10	Não	Não	Não	Não	Não	Não
celular	VARCHAR	10	Não	Não	Não	Não	Não	Não

Observação: Um campo do tipo varchar permite até 255 caracteres, mas o tamanho de campos como rua, por exemplo, deve ir no máximo até 50 caracteres. Isso porque talvez seja preciso emitir etiquetas de endereçamento e como o faremos se o campo rua tiver 255 caracteres? O campo CEP também foi colocado como obrigatório, para que o usuário se lembre de preenchê-lo.

Os campos telefone e celular são do tipo varchar, pois o zero à esquerda é significativo, isto é, 01 é diferente de 1.

FORNECE

nome	tipo de dados	tamanho	obrigatorio	unico	chave primaria	chave estrangeira	valor default	regra de validacao
numero_nota	INTEGER	7	Sim	Sim	Sim	Não	Não	Não
cod_fornecedor	INTEGER	4	Sim	Não	Sim	Sim	Não	Fornecedor já deve ser cadastrado
data	DATE	8	Sim	Não	Não	Não	Não	Não
valor_total	DECIMAL	10,2	Sim	Não	Não	Não	Não	Soma dos itens de itemfornece para esta nota.

ITEMFORNECE

nome	tipo de dados	tamanho	obrigatorio	unico	chave primaria	chave estrangeira	valor default	regra de validacao
numero_nota	INTEGER	7	Sim	Sim	Sim	Sim	Não	Não
cod_produto	INTEGER	5	Sim	Não	Sim	Sim	Não	Produto já deve ser cadastrado
quantidade	DECIMAL	5,2	Sim	Não	Não	Não	Não	Não
valor_unitario	DECIMAL	10,2	Sim	Não	Não	Não	Não	>0

COMPRA

nome	tipo de dados	tamanho	obrigatorio	unico	chave primaria	chave estrangeira	valor default	regra de validacao
numero	INTEGER	7	Sim	Sim	Sim	Não	Não	Auto-Numeração
cod_cartao	INTEGER	8	Sim	Não	Não	Sim	Não	Cartão já deve ser cadastrado
data	DATE	8	Sim	Não	Não	Não	Não	Não
forma_pagto	VARCHAR	10	Sim	Não	Não	Não	Não	Não
valor_total_compra	DECIMAL	10,2	Sim	Não	Não	Não	Não	Soma dos valores dos itens de itemcompra para esta compra
cod_func_caixa	INTEGER		Sim	Não	Não	Sim	Não	Funcionário deve ser cadastrado

ITEMCOMPRA

nome	tipo de dados	tamanho	obrigatorio	unico	chave primaria	chave estrangeira	valor default	regra de validacao
numero_nota	INTEGER	7	Sim	Sim	Sim	Não	Não	Não
cod_produto	INTEGER	5	Sim	Não	Sim	Sim	Não	Produto já deve ser cadastrado
quantidade	DECIMAL	5,2	Sim	Não	Não	Não	Não	Não
valor_unitario	DECIMAL	10,2	Sim	Não	Não	Não	Não	Valor do campo preco_venda da tabela produto quando da inclusão da nota.

Observação: Um campo autonumeração é um campo numérico, que terá seu valor sequencial preenchido pelo próprio SGBD. Mas é importante frisar que nem todo Sistema Gerenciador de Banco de Dados (SGBD) possui esse campo. Por isso, é preciso verificar essa disponibilidade conforme o sistema adotado. Veja que o campo cod_func_caixa não estava definido no projeto lógico, mas, ao analisar as funcionalidades, verificou-se que o senhor João quer saber quem recebeu pela compra e, assim, foi necessário possibilitar o registro desta informação, a qual servirá também para indicar se determinada compra já foi paga ou não.

2.3.8. Projeto físico

O projeto físico consiste na tradução do modelo lógico para a linguagem SQL. Normalmente, é feito um script (lista dos comandos de criação do banco de dados e de suas tabelas dentro do SGBD). Os comandos para gerar as tabelas em SQL serão estudados no próximo capítulo, mas como exemplo, vem a seguir o projeto físico de nosso estudo de caso para o SGBD MySQL.

```
CREATE DATABASE Padaria;
USE Padaria;
```

```
CREATE TABLE Cartao (
codigo INTEGER NOT NULL PRIMARY KEY,
dt_inicio_uso DATE NOT NULL,
dt_fim_uso DATE);
```

```
CREATE TABLE Produto (
codigo INTEGER NOT NULL PRIMARY KEY,
nome VARCHAR(40) NOT NULL,
preco_venda DECIMAL(10,2) NOT NULL,
saldo DECIMAL(10,2) NOT NULL,
estoque_minimo DECIMAL(10,2) NOT NULL);
```

```
CREATE TABLE Funcionario (
codigo INTEGER NOT NULL PRIMARY KEY,
nome VARCHAR(50) NOT NULL,
funcao VARCHAR(30) NOT NULL);
```

```
CREATE TABLE Fornecedor (
codigo INTEGER NOT NULL PRIMARY KEY,
nome VARCHAR(50) NOT NULL,
rua VARCHAR(50) NOT NULL,
```

DICA

A definição das tabelas do modelo lógico deve ser feita com muita atenção e cuidado, pois podemos facilitar, e muito, a implementação da solução, por exemplo, incluindo campos como obrigatórios e definindo regras de inclusão e alteração. Dessa forma, colocamos no SGBD parte da responsabilidade pela consistência dos dados e aliviamos os programas que farão a entrada e a manipulação de dados.

```
complemento VARCHAR(50),
bairro VARCHAR(40),
cidade VARCHAR(40) NOT NULL,
estado VARCHAR(2) NOT NULL,
cep VARCHAR(8) NOT NULL,
telefone VARCHAR(10),
celular VARCHAR(10));
```

```
CREATE TABLE Fornece(
numero_Nota INTEGER NOT NULL PRIMARY KEY,
cod_Fornecedor INTEGER NOT NULL REFERENCES
Fornecedor(codigo),
data DATE NOT NULL,
valor_Total DECIMAL(10,2) NOT NULL),
PRIMARY KEY (numero_Nota,cod_Fornecedor));
```

```
CREATE TABLE ItemFornece (
numero_Nota INTEGER NOT NULL REFERENCES
Fornece(numero_Nota),
cod_Produto INTEGER NOT NULL REFERENCES
Fornecedor(codigo),
quantidade DECIMAL(5,2) NOT NULL,
valor_Unitario DECIMAL(10,2) NOT NULL,
PRIMARY KEY (numero_Nota,cod_Produto));
```

```
CREATE TABLE compra(
numero INTEGER NOT NULL PRIMARY KEY,
cod_Cartao INTEGER NOT NULL REFERENCES
Cartao(codigo),
data DATE NOT NULL,
forma_Pagto VARCHAR(10) NOT NULL,
valor_Total_Compra DECIMAL(10,2) NOT NULL,
cod_Func_Caixa INTEGER NOT NULL REFERENCES
Funcionario(codigo));
```

```
CREATE TABLE ItemCompra (
numero_Nota INTEGER NOT NULL REFERENCES
Compra(numero),
cod_Produto INTEGER NOT NULL REFERENCES
Fornecedor(codigo),
quantidade DECIMAL(5,2) NOT NULL,
valor_Unitario DECIMAL(10,2) NOT NULL,
PRIMARY KEY (numero_Nota,cod_Produto));
```


2.3.9. Análise funcional

Analisando os requisitos funcionais que encontramos na fase de análise de requisitos, serão escolhidas as rotinas a serem criadas para que todas as funcionalidades de nosso projeto sejam atendidas. Uma rotina do sistema pode automatizar mais de um requisito funcional anteriormente definido. Por exemplo, a digitação de uma nota fiscal de compra de mercadorias para a padaria do senhor João implementará não apenas as tabelas Fornece e ItemFornece, mas atualizará o saldo do produto na tabela Produtos, podendo alterar também o valor do campo preço de venda daquele item. Se analisarmos os requisitos funcionais do sistema proposto para a padaria teremos, pelo menos, as seguintes rotinas:

- **Manutenção de cartões**
- **Manutenção de funcionários**
- **Manutenção de fornecedores**
- **Manutenção de produtos**
- **Registro de produto vendido pelos atendentes**
- **Apresentação da somatória da compra, recebimento da compra e marcação de qual funcionário recebeu determinada compra**
- **Lançamento da Nota Fiscal de Entrada**
- **Controle de acesso do usuário**
- **Opções de seleção de rotinas disponíveis para o usuário do sistema, de acordo com seu nível de acesso**
- **Consulta de preços de produtos.**

Após a definição das rotinas e dos requisitos funcionais que elas implementarão, deve-se separá-las em módulos de acordo com a característica de cada uma. Assim, formam-se grupos de rotinas que implementam requisitos semelhantes, a fim de se criar uma estrutura lógica e funcional de acesso às principais funcionalidades do sistema, permitindo, também, o controle de acesso a tais funcionalidades.

No caso do nosso exemplo fictício, a padaria do senhor João, temos alguns grupos de requisitos que tratam da manutenção das tabelas cadastrais do sistema – as tabelas de produto, funcionário, cartão e fornecedor. Podemos nominar as funcionalidades comuns a essas rotinas como, por exemplo, a manutenção de informações cadastrais. Pode-se dizer que os requisitos de registrar um produto em um cartão, lançar uma nota fiscal de um fornecedor, ou registrar o pagamento de uma venda descrevem a movimentação da padaria, logo cabe incluí-los no grupo de rotinas de movimentação do sistema da padaria. Haverá, ainda, grupos para as consultas e relatórios do sistema, nos quais serão reunidas as rotinas que implementarão essas funcionalidades.

A engenharia de software oferece várias técnicas para separar as rotinas do sistema em módulos, como o Diagrama Hierárquico de Funções ou os Diagramas de Pacotes e de Componentes, que veremos no capítulo 4.

2.3.10. Projeto de programas da aplicação

A partir de agora, vamos utilizar vários conceitos relacionados ao desenvolvimento de software. Primeiro será preciso definir a linguagem de programação que adotaremos para implementar a solução. Para isto, temos de levar em conta o ambiente em que o sistema será implementado, ou seja, o sistema operacional

instalado. Também é preciso considerar a estrutura de hardware definida para a solução: se o programa será implantado em uma rede de computadores, qual é sua arquitetura e onde ficarão instalados a aplicação e o Sistema Gerenciador de Banco de Dados (SGDB), tema, aliás, que veremos mais adiante.

Definir um padrão para a interface com o usuário (isto é, das telas a serem exibidas pelo sistema), assim como fontes, cores, formato e tamanho dos elementos das telas (como botões, barras de menu, caixas de texto etc.), requer o conhecimento prévio de detalhes da estrutura em que nossa solução será implementada. Nem sempre os diversos ambientes nos permitem trabalhar com todos esses recursos.

É interessante definir essa interface com o usuário e para isso geralmente usamos a técnica de prototipagem, definida no capítulo 1. Assim, montamos um pequeno protótipo só com a tela a ser exibida e o apresentamos ao cliente para que ele tenha uma ideia de como a visualizará e de como poderá interagir com as informações que solicitou.

É o momento, então, de definir o funcionamento de cada programa a ser criado para a implementação da solução. Também para essa tarefa existem várias técnicas, tais como a descrição em português estruturado do funcionamento de cada programa, o uso de diagramas da UML (diagrama de classes, de sequência, e de máquina de estados, que serão vistos no capítulo 4 deste livro). O que deve ficar claro é como o programa será construído para implementar a rotina definida, em relação à interface com o usuário e ao seu próprio funcionamento, a conexão com o sistema gerenciador de banco de dados e a arquitetura de hardware e software onde a solução será instalada.

Devemos definir também o plano de testes a que devem ser submetidos cada um dos programas em vias de criação, a fim de validar seu correto funcionamento.

2.3.11. Implementação da Transação

Com as definições em mãos, partimos para a programação e os testes das rotinas na linguagem definida. À medida em que essas rotinas vão sendo concluídas, devemos elaborar um ambiente de testes que simule o meio do usuário para que se possa analisar o funcionamento do sistema como um todo, isto é, com todos os programas em funcionamento. Nesse momento, é importante certificar-se de que todos os programas estão executando suas tarefas corretamente, de que o banco de dados está sendo atualizado de forma consistente e de que a estrutura de hardware e software está permitindo o funcionamento correto do sistema e oferecendo ao usuário as informações que ele solicitou, no tempo que precisa delas.

Na fase seguinte, devemos iniciar o treinamento dos usuários, para que possam operar o sistema. No caso da padaria do senhor João, cada atendente tem de saber como se identificar no sistema e como registrar as compras dos clientes no balcão. Os caixas têm de aprender como se registra o recebimento de uma compra e como se lança notas fiscais de fornecedores. E, claro, seria preciso treinar o senhor João, quanto a todas as tarefas do sistema. Tanto no exemplo fictício como em um projeto real, quando todas as informações estiverem cadastradas e os cadastros de funcionários, produtos, cartões e fornecedores estiverem corretos, é hora de estudar a melhor data para a implantação, caso ainda não tenha sido definida. É importante, ainda, estabelecer uma rotina de cópia das informações do sistema (backup). O usuário responsável por essa tarefa tem de ser treinado e instruído sobre sua importância, pois com esse recurso se reduzem as perdas de informação, no caso de falhas de hardware ou software. Na data da implantação e nos dias subsequentes é fundamental acompanhar o funcionamento do sistema, efetuando pequenos ajustes, caso seja preciso. Terminada essa fase, podemos concluir que o sistema está entregue.

Informática, psicologia, administração

Você aprendeu a usar a metodologia para projetar, construir e implementar um sistema. Tarefa cumprida? Ainda não. O trabalho só estará completo quando, além da informática, você lançar mão de seus conhecimentos em psicologia e administração, para lidar com as pessoas envolvidas no projeto e suas expectativas – o que, convenhamos, não é algo simples. Ao projetar e construir um sistema, é preciso ter sempre em mente que os usuários estão ansiosos por vê-lo funcionando em seu ambiente. É possível usar diversas ferramentas para que sua execução seja transparente, o que ajuda a amenizar as expectativas. Pode-se, por exemplo, adotar um cronograma que lhes permita acompanhar todas as fases do projeto, de modo que percebam claramente quais passos já foram dados e os que virão na sequência. Esse, porém, é apenas um dos exemplos de ferramentas auxiliares. O importante é manter a disposição de pesquisar outras, para que o desenvolvimento de nossos projetos se torne cada vez mais rápido e claro.

TETRA IMAGES/CORBIS/LATINSTOCK

Figura 36 Execução de um projeto.