Uma Revisão de Conceitos de Banco de Dados

Geraldo Xexéo

29 de março de 2022 17:15

Sumário

1 Modelos e Abstrações

> The purpose of abstraction is not to be vague, but to create a new semantic level in which one can be absolutely precise
>
> *(Edsger Dijkstra)*

Conteúdo

Por que modelos e abstrações?

Todas as práticas deste livro realização alguma forma de abstração do problema ou solução que fornece modelos aos desenvolvedores e outras partes interessadas.

Este capítulo faz uma introdução aos temas correlatos **modelo** e **abstração**, que são as bases de todos os métodos de Análise de Sistema.

Todas as técnicas estudadas nesse livro implicam na criação de um modelo, seja ele do domínio da aplicação, do negócio, do problema, ou do sistema que será implementado, inclusive modelos que fazem a relação entre modelos distintos.

Apesar de usarmos estes termos de forma coloquial, é importante responder às perguntas: o que é um modelo? O que é uma abstração?

Um **modelo** é uma representação de algum objeto, conceito, conjunto de conceitos ou realidade. Modelos são criados para que nós possamos estudar, normalmente segundo algum aspecto escolhido, o objeto modelado. Na grande maioria das vezes, um modelo é uma versão simplificada, ou seja, abstrata do objeto modelado. Essa versão simplificada permite uma comunicação com foco em alguns conceitos, evitando o que é irrelevante(Yourdon, 2006).

Abstração é o processo mental de separar um ou mais elementos de uma totalidade complexa de forma a facilitar a sua compreensão por meio de um modelo, eliminando (ou subtraindo) o resto. Segundo Rosen (2018), a "abstração é um processo mental distinto em que novas ideias ou conceitos são formadas pela consideração de vários objetos ou ideias e omitindo características que os distingue."

A Tabela 1.1 mostra uma sequência de imagens, a partir de uma foto, que ilustram o conceito de abstração. Enquanto a primeira foto[1] é muito detalhada, as seguintes continuam de alguma maneira representando o conceito de mulher, porém de uma forma cada vez mais abstrata, até que se chega em um ícone que não tem nenhuma relação com a imagem de uma mulher, o espelho de vênus, reconhecido internacionalmente como símbolo do sexo feminino.

Tabela 1.1: Sequência de imagens que mostram abstrações crescentes de uma imagem inicial que representa uma mulher. Foto por Michael Jatremski.

1.1 Modelos

No nosso dia a dia nos deparamos constantemente com modelos. Um mapa é um modelo do território que descreve. Uma maquete de um prédio é um modelo que nos permite ver o prédio a ser construído como um objeto tridimensional. Uma receita de bolo é um modelo do processo para construir o bolo. Mesmo uma foto de um modelo pode ser vista como outro modelo.

Um modelo deve ser simples o bastante para ser fácil de manipular e, simultaneamente, complexo o suficiente de forma a servir para a solução do problema em questão, de acordo com o ponto de vista desejado. Assim, um mapa para navegação tem informações

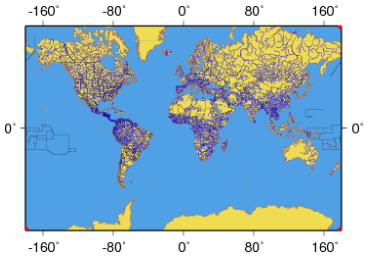
[1]Foto original por Michael Jatremski, https://openphoto.net/gallery/image/view/5539

diferentes de uma mapa para estudo do clima, e um modelo de comportamento do software também tem informações diferentes do modelo da informação que o software armazena.

Quanto mais simples o modelo, maior a abstração feita para produzi-lo, ou seja, mais características são suprimidas do objeto original. Chaitin (2006) chega a dizer que se uma teoria, que é também um modelo, é do mesmo tamanho que o dado que ela explica, então não tem valor nenhum. Ainda segundo o autor, uma teoria só é útil quando é uma compressão dos dados, e compreender é comprimir.

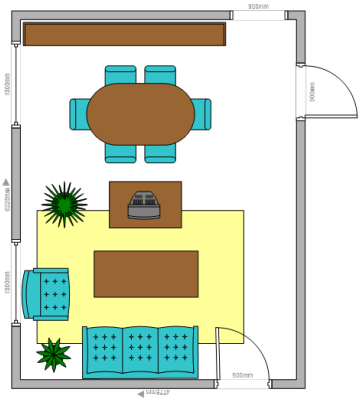
No nosso dia a dia utilizamos continuamente a capacidade humana de abstrair para poder trabalhar com toda a informação que o mundo nos fornece. Voltando ao caso do mapa: ele é um modelo de uma região. Dependendo da informação que queremos, colocamos alguns símbolos e tiramos outros do mapa. Um mapa também não pode ser perfeito, tem que "abstrair" as informações que não são necessárias para a utilização que foi construído. Se não houvesse nenhuma abstração, o mapa teria que ter o mesmo tamanho da cidade.

Podemos usar mapas com diversos graus de detalhe, ou seja, mais ou menos abstratos. Um globo terrestre, por exemplo, é um mapa muito abstrato, geralmente com o objetivo de ensinar noções básicas de geografia. Já uma carta náutica é um mapa muito detalhado, que permite a navios ou barcos menores navegarem de forma segura em uma região. Ainda mais detalhada será a planta de uma casa ou prédio. Os níveis de detalhe são infinitos e são usados de acordo com a necessidade do problema a ser resolvido. A Figura 1.1 exemplifica diferentes mapas, com diferentes níveis de abstração adequados ao seu uso.



(a) Mapa Mundi

(b) Guia de Localização



(c) Planta baixa

Figura 1.1: Diferentes tipos de mapas, ou seja, modelos, cada um com um nível diferente de abstração e dedicado a uma utilização distinta.

Um tipo especial de modelo é o protótipo. Um protótipo é um modelo exemplar, no sentido que fornecer um exemplo, onde as simplificações de certo aspecto são muito pequenas ou não existem. Por exemplo, um protótipo de um software pode ter todas as suas telas na versão final, sendo esse o aspecto estudado, mas nenhuma funcionalidade.

Em algumas áreas, onde há a necessidade de fabricar algo em série, protótipos são um modelo de fabricação, completo em funcionalidade, mas que não foram feitos pelo processo final esperado no momento em que serão produzidos em série.

Finalmente, relativo a aplicação de modelos em nosso problema Yourdon (2006) afirmou que o analista de sistema usa modelos para:

- dar foco as características importantes do sistema e diminuindo a influência de características menos importantes;
- discutir mudanças e correções nos requisitos do usuário a um custo baixo e risco mínimo, e
- verificar que o analista de sistema entende corretamente o ambiente do usuário e o documentou de forma que os outros desenvolvedores possam construir o sistema.

Um típico modelo usado em análise de sistemas, na atualidade, é o Digrama de Casos de Uso, como o apresentado na Figura 1.2.

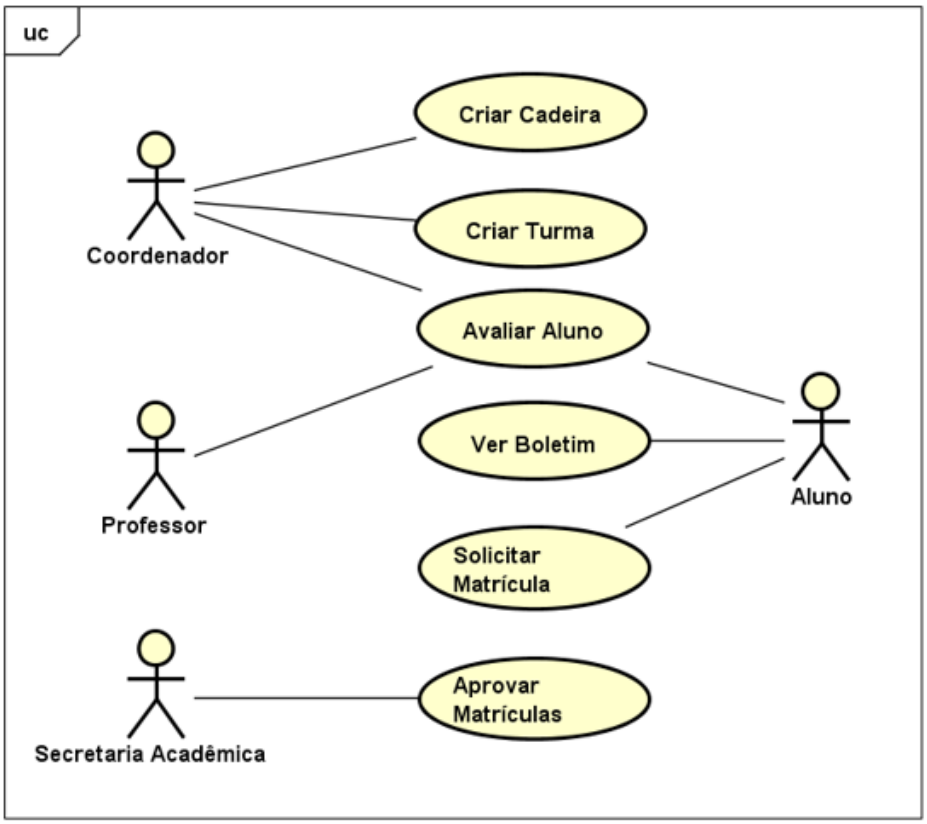


Figura 1.2: Exemplo de Diagrama de Casos de Uso.

1.2 Tipos de Abstrações

Existem várias formas de abstração praticadas no dia a dia ou em situações especiais. No desenvolvimento de sistemas, utilizamos alguns processos de abstração típicos:

- **ocultação da informação** (ou abstração da caixa-preta, ou encapsulamento);
- **classificação**, que relaciona instâncias e classes;
- **generalização**, ou herança, que relaciona classes;
- **composição**, ou todo-parte;
- **identificação**;
- **simplificação pelo Caso Normal**;
- **foco/inibição**;
- **refinamento sucessivo**, e
- **separação de interesses**.

1.2.1 Ocultação da Informação

Pela abstração de **Ocultação da Informação**, deixamos de nos preocupar com o interior de uma coisa, só prestando atenção a seu exterior observável.

Podemos encontrar esse exemplo facilmente em muitas coisas do mundo real. Quantas pessoas, por exemplo, sabem como funciona um telefone celular? Poucas, certamente. Porém quase todas sabem usá-lo, porque "abstraem" o seu funcionamento interno (isso é, não pensam nisso) e colocam em foco apenas o comportamento externo.

Por isso chamamos também essa abstração de **abstração de caixa-preta**. O conceito inverso (ou seja, abrir a caixa) é chamado de **caixa-branca**.

Na área de Computação é muito comum ocultar informação para facilitar a compreensão. Por exemplo, em programação, se você usa uma biblioteca de funções, não está interessado em saber como uma função é realizada, mas sim apenas nos parâmetros de entrada e em como será o resultado da função. A função funciona como uma caixa-preta.

1.2.2 Classificação

A **classificação** é um mecanismo básico do raciocínio humano. Talvez seja um dos que mais nos habilita a tratar de toda informação que recebemos diariamente.

No processo de classificação eliminamos parte da individualidade do objeto ou sistema analisado e o consideramos como um exemplar de uma **classe**. Quando fazemos isso, aceitamos que esse objeto, agora uma **instância** da classe, divide com todas as outras instâncias da classe um conjunto de características.

Segundo Parrochia (2020), **classificar** é "a operação de distribuir objetos em classes ou grupos, que são, em geral, menos numerosos que eles.". Essa operação simplifica o mundo, tendo tem vantagens principais Parrochia (2020):

- substitui um multiplicidade caótica e confusa por um ordem racional e regular;
- permite trabalhar com um número reduzido de classes de equivalências em vez de trabalhar com os elementos, e
- detectam uma simetria nas classes que diminui a complexidade do problema e simplifica o mundo.

Na classificação o que estamos fazendo é imaginar uma ideia única que descreve, de forma abstrata, todos os objetos de uma conjunto. Ao eliminar a necessidade de tratar cada objeto de forma única, simplificamos o problema em questão.

A classificação é uma relação entre **instâncias e classes**. É possível associar uma classe com um conjunto de elementos que possuem algumas de propriedades em comum, como todos os alunos de uma universidade ou todos os animais mamíferos.

O processo reverso da classificação é a **instanciação**. O conjunto de todas as instâncias de uma classe é a extensão dessa classe. Uma classe pode ser descrita por sua extensão, ou por sua intenção, que é uma descrição que permita identificar um conjunto de entidades(Parrochia, 2020).

Exemplos típicos de classificação aparecem na Tabela 1.2.

Tabela 1.2: Exemplos de Classificação

Instâncias	**Classe**
Flamengo, Fluminense, São Paulo	Times de Futebol
Brasil, Estados Unidos	País
Pelé, Zidane, Messi	Jogador de Futebol

As classificações mais famosas certamente são a Classificação Científica, que classifica todos os seres vivos, e a Classificação Decimal Universal (CDU), que classifica documentos e é utilizada nas bibliotecas.

É importante notar que na vida real um objeto pode pertencer a várias classes. Uma pessoa pode ser um aluno, um professor, um policial, etc... Normalmente, em modelos teóricos como os que vamos usar, tentamos com que um objeto pertença, diretamente, a só uma classe, de modo a facilitar a manipulação do modelo.

Nada impede que uma classe seja uma instância de outra classe, que pode ser chamada de uma **meta-classe**, de uma outra hierarquia. Em Smalltalk-80, por exemplo, toda classe é uma instância da classe *Class*. Também podemos analisar que **Homo** é uma classe que é uma instância da classe *Gênero*, que reúne todos os gêneros. Na prática a classe "Gênero" e as outras similares, como "Espécie" são meta-classes no modelo da Classificação Científica. Não podemos confundir, porém, essa relação com a próxima, a de generalização.

As classificações também são uma forma especial de Ocultação da Informação, pois trabalhando com a classe ocultamos as informações como a identidade.

1.2.3 Generalização

Com a **generalização** nós somos capazes de entender como uma classe pode ser descrita por outra classe, mais geral. É importante ver a **diferença entre a classificação e a generalização**: a primeira trata da relação entre objeto e classes, enquanto a segunda trata da relação entre classes.

A generalização é uma relação muito comum entre classes, permitindo que qualquer objeto de uma classe possa ser visto, de uma forma mais geral, como um objeto de outra classe. Por exemplo, "Alice" é uma instância da classe "mulher", enquanto "Bruno" é uma instância da classe "homem", porém ambas as classes, "homem" e "mulher" podem ser generelizadas na classe "humano". Dessa forma, "Alice" e "Bruno" deve ser tratados similarmente enquanto membros da classe, mais geral, "humano", mas podem ser tratados de foram diferente enquanto membros das classes, mais específicas, "mulher" e "homem". Utilizando judiciosamente a generalização podemos simplificar a forma de tratar objetos de classes similares em uma hierarquia.

O processo reverso da generalização é a **especialização**. Ela é uma relação entre classes, ao contrário da classificação que é uma relação entre classes e instâncias.

Exemplos típicos de generalização aparecem na Tabela 1.3.

Tabela 1.3: Exemplos de Generalização

Classes	**Classe mais geral**
Funcionário, Aluno, Professor	Pessoa
Automóvel, Avião, Navio	Meio de Transporte
Computador, Rádio, Televisão	Aparelhos Eletrônicos

Novamente a Classificação Científica dos seres vivos nos oferece um grande exemplo da relação de generalização. Cada instância da Espécie *Homo sapiens* é uma instância da Espécie *Homo*, logo a classe *Homo* é uma generalização da classe *Homo Sapiens*.

Na linguística a relação de generalização é conhecida como **hiperonímia**, e a especialização é conhecida como **hiponímia**.

1.2.4 Composição ou Agregação

Na **composição** entendemos um objeto complexo, formado de um conjunto de outros objetos, como um só objeto. É como vemos uma bicicleta ou um carro. Ao eliminar a necessidade de descrever as partes, simplificamos a compreensão do objeto analisado.

O processo reverso da composição é a **decomposição**. A Figura 1.3 mostra um brinquedo, o todo, e as partes que o compõe. As partes em separado não são o brinquedo.

(a) Partes do brinquedo (b) Brinquedo completo

Figura 1.3: Um brinquedo é composto de suas partes. Foto do autor

Exemplos típicos de composição aparecem na Tabela 1.4.

Normalmente, em modelagem de dados, usamos o conceito de composição para dizer que uma classe (como endereço) é uma característica de outra classe, descrevendo um entre seus atributos.

Tabela 1.4: Exemplos de Composição

Partes	**Objeto**
Pneus, motor, etc.	Carro
Capa, centenas de folhas, etc.	Livro
Cabelo, pele, ossos, etc.	Homem

Na linguística a parte é conhecida como um **merônimo** e o todo como um **holônimo**. As relações são de **meronímia** ou **holonímia**.

1.2.5 Identificação

Com a **identificação** nós somos capazes de entender como caracterizar unicamente um objeto. Um nome identifica uma pessoa, por exemplo. Ao identificar unicamente um objeto podemos separá-lo de outro objeto semelhante e atribuir a entidades específicas atributos e características que só pertencem a ela, e não pertencem a outros elementos daquela classe.

Há uma diferença entre instanciar e identificar. Uma instância deve possuir uma identificação e uma identificação se aplica a uma instância. A identificação permite a que duas instâncias sejam reconhecidas como distintas ou como representações de um mesmo objeto (normalmente devendo ser reunidas em uma).

1.2.6 Simplificação pelo Caso Normal

Toda aplicação, ao funcionar, deve tratar de casos específicos que ocorrem durante o funcionamento normal, sejam eles alternativas que são aceitas no negócio, como alternativas não aceitas ou mesmo erros do sistema. Porém, é bem mais fácil discutir o funcionamento normal antes e depois os casos específicos.

A abstração de **simplificação pelo caso normal** indica que devemos começar a trabalhar pelo modo comum ou normal de funcionamento, ou ainda melhor, o modo onde tudo ocorre da forma mais simples e depois ir inserindo mecanismos para tratar das variações possíveis, por meio da especificação das **condições** que levam a essas variações.

1.2.7 Foco e Inibição

Uma das características importantes do ser humano é ser capaz de prestar atenção, isto é, por o foco em um detalhe, inibindo parcialmente os outros detalhes ao redor, e assim processar a informação, detalhe a detalhe.

Podemos ver essa forma de abstração acontecer no dia a dia, quando estamos olhando para um local e as áreas ao redor ficam levemente vigiadas pela visão periférica, mas não estamos realmente prestando atenção nas mesmas.

A abstração de simplificação pelo caso normal é uma forma de colocar o foco em uma questão e inibir as outras, i.e., as variações, que depois serão o foco da análise uma a uma, mantendo as outras inibidas.

Isso também acontece em um modelo de dados. Cada parte de um modelo foca em alguma informação que pretendemos registrar, e possui regiões ao redor que nos informam outras informações adicionais, mas não precisamos olhar ao detalhe da outra parte para entender aquela. Por isso separamos o sistema em entidades, como “aluno”, “curso” e “professor”, e depois analisamos cada uma em separado.

Tecnicamente falando, foco e inibição são muitas vezes representados pela modularização e divisão do sistema em partes estanques, com as características de alta coesão, todo um módulo trata apenas de um assunto, e baixo acoplamento, um módulo não interfere no outro. Outra forma de usar o foco e a inibição é no refinamento sucessivo.

1.2.8 Refinamento Sucessivo

A técnica de **refinamento sucessivo** indica que cada problema deve ser tratado de uma forma mais geral para depois ser analisado de uma forma mais específicas, normalmente seguindo o conceito de “explodir” um problema anterior (mais geral) em sub-problemas mais específicos, sendo cada um desses sub-problemas “explodidos” também até chegarmos a um problema de solução simples.

O refinamento sucessivo é uma forma mais específica da abstração de Foco e Inibição e faz parte de várias estratégias de abstração baseadas no conceito de **dividir para conquistar**. Equivale a uma estratégia *top-down* de solução de problemas, cuja a inversa é a estratégia *bottom-up*

1.2.9 Separação de Interesses

Separação de interesses é o processo de abstração onde tentamos descrever, ou produzir, um conceito separando-o em conceitos distintos com a menor quantidade possível de interseção, baseado em algum aspecto específico do problema sendo tratado. Esses conceitos normalmente são características ou comportamentos.

A separação de interesses é uma forma mais específica da abstração de Foco e Inibição e também faz parte de várias estratégias de abstração baseadas no conceito de dividir para conquistar.

1.3 Trabalhando com as abstrações

Imagine que precisamos descrever comprar um carro. É óbvio que todo carro possui quatro pneus, um motor, etc. Isso é uma classe bastante geral. Porém, desejamos ainda falar sobre um modelo específico: uma Ferrari Testarossa, por exemplo. Logo, acabamos de especializar nosso modelo, mas ainda não chegamos ao nível de objeto. Quando vemos o carro específico, aí temos o objeto. Ele é identificável como instância daquela classe porque apesar de dividir várias características em comum com outros objetos da classe, também tem algumas características únicas, como o número de série do chassi. Finalmente, desejamos trocar a cor do assento do carro. Nesse instante, já estamos vendo uma parte do carro, decompondo-o em suas partes.

1.4 Exercícios

Exercício 1.1: Vá para o site `http://jogodeanalisedesistemas.xexeo.net/` e visite a Livraria Resolve. Faça um modelo na forma de um diagrama de toda a informação levantada. Pode ser, por exemplo, como um Mapa Mental.

2 Dados e Informação

The goal is to turn data into information, and information into insight.

(Carly Fiorina)

Conteúdo

Este capítulo trata de quatro palavras que no dia a dia parecem ser sinônimas, mas têm significados diferentes quando usadas na Análise de Dados: dado, informação, conhecimento e sabedoria.

Vamos recorrer a dicionários para ter uma definição inicial. Segundo o American Heritage, **informação**, no contexto de computadores, "é o dado quando processado, guardado ou transmitido"(American Heritage, 2019).

Já no dicionário Aurélio, informação, entre outros significados, pode ser "Conhecimento amplo e bem fundamentado, resultante da análise e combinação de vários informes", "Coleção de fatos ou de outros dados fornecidos à máquina, a fim de se objetivar um processamento" ou ainda "Segundo a teoria da informação, medida da redução da incerteza, sobre um determinado estado de coisas, por intermédio de uma mensagem".

Apesar de não estarmos diretamente envolvidos com a teoria da informação, não podemos de deixar de notar a importância da definição que diz que a **informação reduz a incerteza por meio de uma mensagem**.

Na teoria da informação a quantidade de informação de uma mensagem pode ser medida como o menor número de bits para representar ela completamente.

2.1 Usando Significados Diferentes

Dados, informação, conhecimento e sabedoria[1] são quatro palavras que na língua corrente possuem uma interseção de significados, porém são consideradas diferentes em áreas específicas, como Computação e Engenharia de Sistemas e Sistemas de Informação.

A distinção entre essas palavras e a existência de uma hierarquia de entendimento entre elas é uma discussão corrente, construída em cima de uma pirâmide que registra a complexidade do que esses termos significam, conhecida como hierarquia DIKW (Rowley, 2007).

Diversos autores propõe definições diferentes. Zins (2007) chega a apresentar 44 definições diferentes para dados, informação e conhecimento, fornecidas por especialistas no assunto reunidos em um painel, usando a metodologia *Critical Delphi* (Zins, 2012).

Seguindo a classificação de Zins (2007), preferimos uma conceituação baseada em um modelo onde dados e informação são fenômenos externos, enquanto conhecimento, e consequentemente sabedoria, são fenômenos internos. Isso significa que dados e informação são do domínio universal, externos a mente, enquanto conhecimento e sabedoria são do domínio subjetivo, internos a mente(Zins, 2007).

Dados são apenas os símbolos que usamos para representar a informação, o registro de diferentes aspectos de um fato ou fenômeno, como Os números que guardamos em um banco de dados. Dados não são interpretados, eles existem, são adquiridos de alguma forma, via coleta, pesquisa ou criação, guardados de outra forma e, possivelmente, apresentados em uma terceira. O computador é uma máquina programável que manipula dados.

Por outro lado, informação é o dado com significado, normalmente processado de forma a ser útil. Uma informação deve permitir responder perguntas como “quando”, “quanto”, “quem”, etc.

$$\text{Informação} = \text{Dado} + \text{Significado} \tag{2.1}$$

É necessário fazer um mapeamento entre dados e informação. Esse mapeamento pode ser simples ou complexo, dependendo de várias variáveis envolvidas, que vão desde decisões arbitrárias tomadas pelo desenvolvedor até padrões internacionais. Por exemplo, em muitos sistemas é preciso ter a informação do sexo de uma pessoa (masculino ou feminino). Para isso, são guardados um número (1 ou 0) ou uma letra (M ou F), que é o dado que faz o registro da informação.

Existem outras definições para dados e informação, mas esta diferenciação entre símbolo e significado será suficiente neste livro. Em especial, na Teoria da Informação tem uma abordagem onde o significado não é importante, apenas a quantidade de mensagens possíveis(Claude E Shannon, 1963).

[1] Em inglês, *wisdom*

Definir conhecimento e sabedoria é muito mais complexo, sendo conceitos em que há mais incerteza ou mesmo desacordo sobre o que realmente significam formalmente, além do significado ambíguo dessas palavras na língua portuguesa ou inglesa. Porém, vamos tomar algumas decisões sobre esses significados, sempre no contexto deste livro.

Sumarizando vários autores, Rowley (2007) define conhecimento como "uma mistura de informação, entendimento, capacidades, experiências, habilidades e valores." Além disso, o mesmo autor lembra a diferença entre conhecimento explícito, que está codificado em documentos, bancos de dados e outros registros, e tácito, que é está na mente das pessoas, ou nos processos que elas realizam. Essa diferença foi bastante explorada por Nonaka e Takeuchi (1995), que apontam também a dificuldade de processar computacionalmente conhecimento tácito.

Finalmente, Rowley (2007) afirma que vários autores tratando de dados, informação e conhecimento não falam de sabedoria, e aponta uma publicação anterior sua, que define sabedoria como "a capacidade de por em ação o comportamento mais apropriado, levando em conta o que é conhecido (conhecimento) e o que traz o bem maior (considerações sociais e éticas)."(Rowley, 2006).

Rowley (2007) também nos lembra que enquanto dados são muito fáceis de processar e contém pouco significado, e ficam na base da hierarquia, sabedoria é muito difícil de processar e fica no topo da mesma. Isso resulta em uma pirâmide como na Figura 2.1.

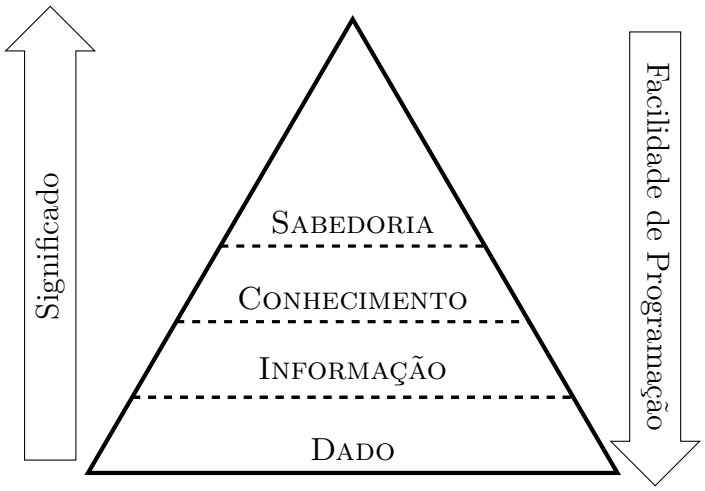


Figura 2.1: Pirâmide da hierarquia do DIKW. Fonte:(Rowley, 2007)

2.2 Conclusão

Por exemplo, um sistema pode ter conter milhares de registros representando vendas, com números guardados em um formato binário que nada significa para o usuário, mas o usuário vai poder ver em um relatório o valor total vendido no mês, informação que é relevante para ele.

O conhecimento normalmente está mapeado nos algoritmos do sistema, nas regras de negócio implementadas. Alguns sistemas, que por exemplo usam aprendizado de máquina, podem inferir novos conhecimentos. Um exemplo disso seria o sistema de vendas descobrir que um tipo de item é mais vendido no início do mês, ou sempre junto de outro item, o que é chamado de análise de cesta de compras.

Já a sabedoria parece, por enquanto, estar ainda nas mãos dos seres humanos.

3 Modelo de Entidades e Relacionamentos

Conteúdo

3.1 Modelo Conceitual

O objetivo da modelagem conceitual é fornecer aos desenvolvedores uma descrição abstrata de alto nível, independente de tecnologia, da informação contida no sistema. Essa descrição é conhecida como o esquema conceitual da base de dados.

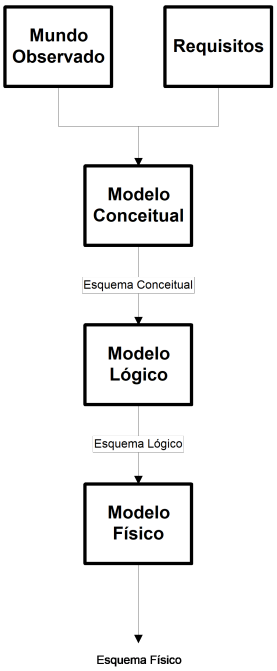


Figura 3.1: As três camadas de modelo de dados.

A Modelagem Conceitual de Dados pode ser feita de muitas formas, algumas vezes com sutis diferenças. Alguns autores defendem a "modelagem do domínio", onde tentamos descrever o domínio de aplicação sendo utilizados, outros tratam diretamente do sistema sendo desenvolvido.

A principal forma de construir um modelo conceitual de dados, aplicado a dados estruturados e SGDB, é usar o Modelo de Entidades e Relacionamentos (MER)(Bertini, Ceri e Navathe, 1992), na forma do Diagrama de Entidades e Relacionamentos (DER), ambos formando o tema deste capítulo. Outro modelo comum é o Modelo Dimensional (Kimball et al., 2008).

Um dos subsídios mais importantes para a criação do DER é o conjunto de regras de negócio levantadas. Muitas das regras de negócio são representadas diretamente no modelo conceitual. Veremos mais tarde que termos e fatos são candidatos naturais para serem objetos nos nossos modelos conceituais. Não devemos confundir, porém, regras de negócio com modelos de dados. Um relacionamento em uma regra de negócio pode representar uma função do sistema, enquanto um relacionamento no MER representa algo que deve pertencer à memória do sistema.

3.1.1 Modelo Lógico

O modelo lógico descreve a informação contida no sistema de acordo com uma tecnologia adotada, sem utilizar, porém, detalhes de implementação. Ele descreve a estrutura do banco de dados que será processado por um SGDB.

Atualmente, o modelo mais utilizado é o modelo relacional. Além dele, alguns modelos distintos podem ser encontrados em aplicações especiais, como data-warehousing e sistemas de informação geográfica.

3.1.2 Modelo Físico

No modelo físico devemos levar em conta não só a tecnologia sendo utilizada, mas também os produtos específicos e a interação do sistema com o ambiente de desenvolvimento e operação. É nessa etapa que nos preocupamos com as principais questões de desempenho, como escolha de índices, particionamento, etc.

3.2 Modelo de Entidades e Relacionamentos

O **Modelo de Entidades e Relacionamentos** (MER), segundo Cougo (1999), descreve o mundo como: "...cheio de coisas que possuem características próprias e que se relacionam entre si". Essas descrições são feitas sobre mundos restritos, e se referem a informação necessária para que um sistema de informações cumpra suas funções. Normalmente a descrição é feita na forma de um diagrama, o diagrama de entidades e relacionamentos.

Existem várias propostas de representação do MER. Em quase todas elas, no diagrama de entidade e relacionamentos cada tipo de entidade é representado por um retângulo, identificado pelo nome da entidade.

A Figura 3.2 mostra um diagrama de entidades e relacionamentos simples, usando uma evolução da notação original de Chen (1990) proposta por Bertini, Ceri e Navathe (1992). Esse diagrama, que descreve um mini-mundo sobre novelas, mostra as entidades Novela, Capítulo, Ator, Diretor, Ator Horista e Horas. Ele também apresenta os relacionamentos Dirige, Compõe, Atua, Pode ser e Trabalha, que ligam, cada um, duas dessas entidades. Os números indicam a cardinalidade da participação das entidades nos relacionamentos. Assim, a leitura do modelo da Figura 3.2 é:

- Entidades do modelo: Diretor, Novela, Capítulo, Ator, Ator Horista e Hora
- Um diretor dirige no máximo uma novela, podendo não dirigir nenhuma, e uma novela é dirigida por um e apenas um diretor.
- Um capítulo compõe uma e apenas uma novela e uma novela tem no mínimo um capítulo, podendo ter um número ilimitado deles.

- Um ator atua em uma novela, podendo não atuar em nenhuma e uma novela tem ao menos um ator, podendo ter vários.
- Um ator pode ser um ator horista e um ator horista é obrigatoriamente um ator.
- O registro do trabalho de ator horista é feito por uma entidade chamada Horas, que indica a quantidade de horas trabalhadas.

Algumas observações podem ser feitas nesse modelo. A primeira é que os capítulos não estão sequenciados. Na prática, algum atributo deverá ser adicionado ou a "Capítulo", ou ao relacionamento "compõe" para caracterizar o número do capítulo. A segunda é que a entidade "Horas" não assume valores, devendo ter atributos que contenham ou permitam calcular a quantidade de horas trabalhadas.

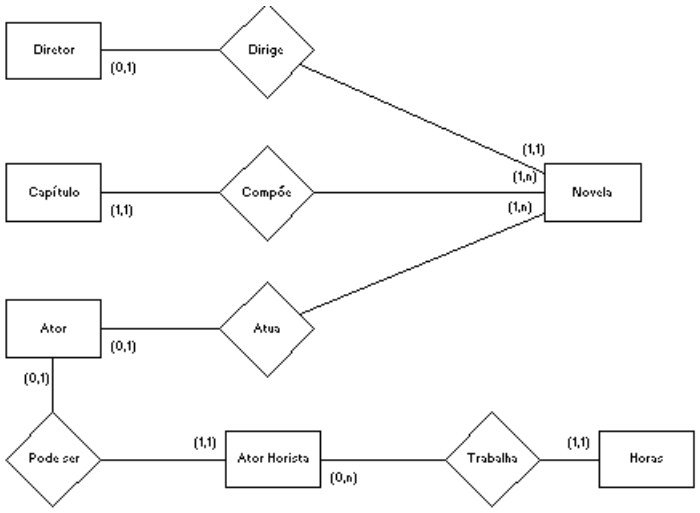


Figura 3.2: Um diagrama de entidades e relacionamentos simples, sem mostrar atributos

As coisas as quais o MER se refere podem ser pessoas, objetos, conceitos, eventos, etc., existentes ou imaginárias. Elas são classificadas em **entidades**. Alguns autores preferem o termo **tipo de entidade** ao termo entidade. Este texto usa os termos entidade e tipo de entidade indiferentemente, para representar a classe.

A priori, só exigimos de uma entidade que cada um dos seus membros possa ser identificado distintamente, isso é, tenha identidade própria. Cada coisa distintamente identificada é uma instância.

Para representar os objetos específicos, os membros da classe, é adotado o termo **instância**, ou **instância de entidade**. Porém, no discurso normal, a palavra entidade também é muitas vezes usada no lugar de instância.

Uma entidade representa uma classe de objetos do universo de discurso do modelo. Por exemplo, em uma universidade podemos encontrar um funcionário chamado funcionário José e uma aluna chamada Maria. José uma instância da entidade funcionário, enquanto

Maria é uma instância da entidade aluna. Funcionário e aluno são os tipos de entidade. Cada instância, então, deve poder ser identificada unicamente.

A classificação consiste em resumir uma quantidade de características comuns de objetos que tem identidade própria por meio da criação de uma classe, ou tipo, que os descreva de alguma forma. Assim, é possível saber que todos os funcionários, por serem instâncias de um mesmo tipo, possuem características comuns (como trabalhar na universidade, ter um salário, etc.).

Matematicamente, entidades, ou tipos de entidade, são conjuntos. Os elementos desse conjunto são as instâncias. Em um modelo, as entidades representam todas as instâncias possíveis. No banco de dados que implementa esse modelo estarão apenas as instâncias que interessam ao sistema.

Apenas algumas entidades do mundo real (ou imaginário) são de interesse para o sistema. Durante a modelagem conceitual nos preocupamos com as "coisas" que o sistema deve lembrar e colocamos essas "coisas" no modelo de entidade e relacionamentos. Uma entidade deve ser relevante para o objetivo do negócio e necessária para a sua operação.

Cada entidade tem dois tipos de características importantes: seus atributos e seus relacionamentos.

Os **atributos** são características que toda a instância de um tipo possui, mas que podem variar entre as instâncias. Uma instância do tipo "aluno" tem os atributos "nome" e "ano de matrícula", por exemplo.

Atributos caracterizam a informação que deve ser guardada sobre uma entidade. Só devemos colocar como atributos aquelas informações que o sistema precisa lembrar em algum momento. Assim, uma instância de "aluno" não precisa ter o atributo "nome do animal de estimação" em um sistema acadêmico, pois apesar de ser algo importante para o "aluno" propriamente dito, não tem importância alguma para o sistema.

A Figura 3.3 mostra um diagrama de entidades e relacionamentos sobre uma locadora de fitas de vídeo[1].

Cada característica deve possuir um **domínio**. O domínio indica o conjunto de valores válidos para a característica. No caso de "nome", geralmente aceitamos qualquer sequência de caracteres, enquanto no caso de "altura" podemos aceitar apenas valores reais positivos menores que 2,5.

Atributos eram originalmente descritos por círculos no modelo E-R. As notações mais modernas anotam os atributos dentro dos retângulos da entidade a que pertencem, como na Figura 3.3.

Finalmente, como indica o nome do modelo, entidades podem se relacionar entre si. Essa característica é a principal força do modelo de entidades e relacionamentos, pois permite que, de certa forma, "naveguemos" no modelo.

[1] O que mostra a idade do autor.

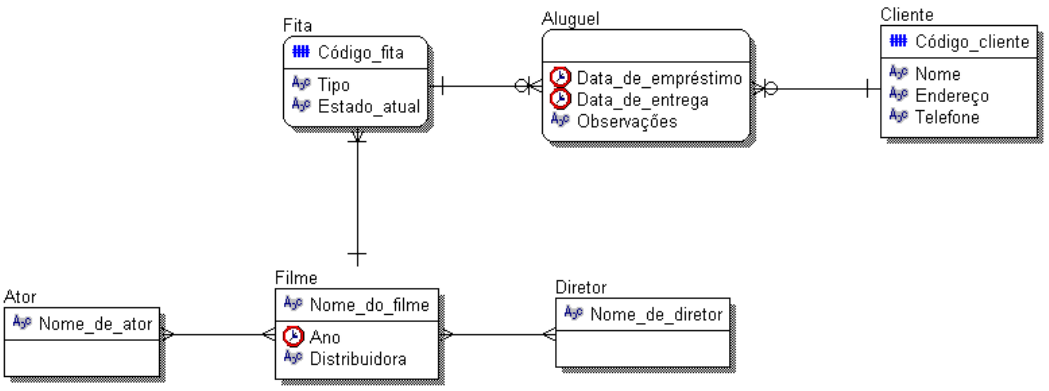


Figura 3.3: Um modelo de entidades e relacionamentos escrito na notação da Engenharia de Informação

Podemos indicar relacionamentos apenas pelas entidades envolvidas, como "cliente-pedido", ou usar um termo como "solicita" que descreva o relacionamento "cliente solicita pedido".

Modelos de Entidades e Relacionamentos para serem completos exigem também um conjunto de restrições. Algumas dessas restrições, como a cardinalidade dos relacionamentos que veremos a seguir, podem ser descritas em algumas (ou todas) notações. Porém, a maioria das descrições é muito complexa para ser descrita em um diagrama. Nesse caso são necessárias anotações ao diagrama descrevendo as descrições. Isso pode ser feito em linguagem natural ou em alguma notação formal específica, dependendo de escolhas da equipe de projeto ou do método utilizado.

3.3 Diagrama de Entidades e Relacionamentos

Existem muitas notações para Diagrama de Entidades e Relacionamentos. A notação original foi proposta por Chen e é composta de entidades (retângulos), relacionamentos (losangos), atributos (círculos) e linhas de conexão (linhas) que indicam a cardinalidade de uma entidade em um relacionamento. Chen ainda propõe símbolos para entidades fracas e entidades associativas.

As notações modernas abandonaram o uso de símbolos especiais para atributos, incluindo a lista de atributos, de alguma forma, no símbolo da entidade. Consideramos as notações mais interessantes na atualidade:

- Engenharia de Informação, também conhecida como *crow´s foot* ou notação de Martin, bastante difundida e também presente como notação alternativa no ERWIN, exemplificada na Figura 3.3;

- Notação de Bertini, Ceri e Navathe (1992), pouco difundida, mas com aspectos teóricos interessantes, ou
- Uso da UML para representar modelos de dados não-orientados a objetos.

Além disso, a notação IDEF1X, utilizada pela ferramenta ERWIN, é bastante difundida no mercado(NIST, 1993);

Toda a notação moderna tem como característica importante definir a cardinalidade mínima e máxima em uma relação, não utilizar um símbolo especial para relacionamentos, mas apenas a linha, e descrever atributos dentro do símbolo de entidade.

3.4 Desenvolvendo um Modelo Conceitual

Desenvolver um modelo conceitual correto para um sistema, completo e consistente, não é uma tarefa fácil. Já desenvolver um modelo conceitual razoavelmente correto, que dê uma idéia do sistema e do negócio e que, de forma evolutiva, resulte em um modelo conceitual correto, é uma tarefa razoavelmente fácil para um analista experiente. Para o analista de sistemas iniciante, porém, parece uma tarefa extremamente difícil.

Isso acontece porque o modelo conceitual de dados exige duas coisas: um alto grau de abstração e a internalização, pelo analista, de conceitos bastante vagos, como "entidades" e "relacionamentos". Assim, o analista iniciante precisa seguir algumas estratégias para entender melhor como desenvolver um modelo conceitual.

A primeira estratégia é a estratégia dos exercícios e exemplos. Nada é tão útil ao aprendizado como colocar a mão na massa. Os exemplos, por seu lado, servem não só como orientação geral, mas também como exemplos de pontos específicos da modelagem e de como especialistas resolvem problemas de modelagem, sejam eles simples ou complicados.

A segunda estratégia é desenvolver uma lista de dicas de trabalho. Essas dicas funcionam para o analista como as pistas funcionam para um detetive, mostrando que caminho seguir até encontrar a solução do problema.

3.5 Entidades

Uma **entidade** é uma pessoa, objeto, local, animal, acontecimento, organização ou outra ideia abstrata sobre a qual o sistema deve se lembrar alguma coisa.

Cada instância de uma determinada entidade tem características similares (mas não iguais), o mesmo comportamento e uma identidade própria.

O primeiro passo na determinação das entidades é o levantamento dos candidatos à entidade. Durante as entrevistas e reuniões de análise de sistema, vários objetos e conceitos serão descritos como parte do sistema. Algumas vezes esses objetos são bastante concretos, como um "produto" dentro de um "estoque", outras vezes são descritos como

documentos que guardam alguma informação, como uma "nota fiscal", outras vezes são abstratos, como um "curso".

No discurso fluente durante uma entrevista, entidades são geralmente substantivos ocupando o papel de sujeito ou objeto, enquanto relacionamentos geralmente são encontrados na forma de verbo. Muitas vezes uma regra de negócio, como "alunos cursam turmas" ou "clientes fazem pedidos", nos indica entidades e relacionamentos .

Outro sinal importante da necessidade de uma entidade é o fato de algo que precisa ser lembrado representar um conceito ou ideia completa. Em um sistema acadêmico precisamos nos lembrar do nome do aluno, da data de matrícula do aluno, do curso em que está o aluno, etc. O "aluno" é a nossa ideia completa que aparece várias vezes, algumas vezes caracterizado por seu nome, outras vezes por seu curso. É um bom candidato a entidade.

Alguns autores propõem uma determinação *bottom-up* das entidades, sugerindo que elas sejam construídas pela partição de todos os dados atômicos que o sistema deve lembrar (nome de aluno, data de matrícula do aluno, etc.). Assim, construiríamos uma lista de atributos para depois agrupá-los em entidades. Preferimos, porém, uma abordagem de busca direta das entidades. Um sistema com poucas dezenas de entidades pode ter centenas de atributos, o que torna a estratégia *bottom-up* mais confusa.

Aproveitando da cultura da orientação a objetos, podemos adotar a divisão proposta por Shlaer e Mellor (1999), uma entidade pode estar em cinco grandes categorias:

- **objetos tangíveis**;
- **papéis exercidos**;
- **eventos**;
- **interações**;
- **especificações**;

Podemos facilmente ver porque objetos tangíveis são bons candidatos a entidades: normalmente, sistemas de informação falam em algum momento de objetos tangíveis, como produtos e equipamentos. Algumas vezes, porém, um objeto tangível, como uma pessoa, assume uma função ou papel específico, como aluno ou professor.

Eventos, ou interações, acontecem em algum momento do tempo e representam classes importantes de entidades. Um exemplo de evento é uma "reunião" em uma agenda . Normalmente eventos exigem atributos como data e duração.

Exemplos típicos de interações são: "contratação de serviço" ou "venda de produto". Interações são semelhantes a relacionamentos ou a objetos tangíveis ou eventos, sendo muitas vezes representadas dessa forma.

Finalmente, especificação são tipos especiais de entidades que classificam outra entidade. Um bom exemplo é "fábrica", que é uma especificação para "automóvel". Geralmente, especificações também podem ser implementadas como um atributo na entidade especificada, sendo essa uma decisão de análise.

Já Coad, Luca e Lefebvre (1999), ao buscarem uma forma mais objetiva de modelar objetos, sugerem que busquemos inicialmente 4 tipos de objetos (que podemos entender como entidades):

- **momentos** ou **intervalos** - cor rosa: um momento ou um intervalo representa qualquer coisa que precisa ser acompanhada, por motivos de negócio ou legais, e que acontecem em um instante de tempo ou por um período de tempo. Muitas vezes pode ser mais fácil começar nossa análise por esse tipo de entidade, pois estamos tratando de atividades de negócio que devem exigir a participação das outras entidades. Exemplos são: aulas, consultas, contratação, etc.
- **papéis** - cor amarela: representam papéis assumidos pelas pessoas que estão envolvidas com o sistema sendo analisado. Cuidado, pois não são apenas os usuários, nem representam os cargos que as pessoas ocupam nas empresas necessariamente. Exemplos são: aluno, professor.
- **pessoas**, **locais** ou **coisas** - cor verde: representam os objetos tangíveis e localidades. Exemplos são: sala, automóvel.
- **descrições** - cor azul: são basicamente as especificações propostas por **DESCOBRIR**. Modelos de um produto é um bom exemplo.

Os quatro estereótipos coloridos do modelo de Coad, Luca e Lefebvre (1999) ainda trazem um padrão de uso que é bastante interessante: relacionamentox entre as pessoas e os intervalos podem ser mediados por um papel da pessoa no intervalo. Assim, um contrato de aluguel é um intervalo, onde uma pessoa é o locatário e outra o locador, ou seja, existe uma entidade pessoa que assume dois possíveis papéis.

A Figura 3.4 mostra um exemplo do uso dessa notação e do padrão descrito.

J. Robertson e S. Robertson (1998) sugerem algumas regras para que verifiquemos se um conceito deve ser realmente escolhido como uma entidade:

- toda entidade deve ter um papel único e definido no negócio, se você não pode explicá-la, provavelmente não precisa se lembrar dela;
- entidades devem ter ao menos um atributo que as descrevam, e é preferível que tenham vários;
- entidades devem ter mais de uma instância. Se a instância é única, então não deve ser uma entidade, mas uma informação constante, que é parte do negócio da empresa (uma regra de negócio?);
- entidades devem possuir instâncias unicamente identificáveis;
- entidades não possuem valores, apenas atributos possuem valores;
- pessoas e organizações que interagem com o sistema são candidatos a entidade quando precisamos nos lembrar alguma coisa específica sobre elas, para gerar relatórios ou processar dados entrados. Isso não se aplica a "logons" ou "passwords" utilizados para a segurança do sistema, pois segurança é um problema tratado no projeto físico. Devemos aplicar essa regra em relação à necessidade de identificação e endereçamento, por exemplo;

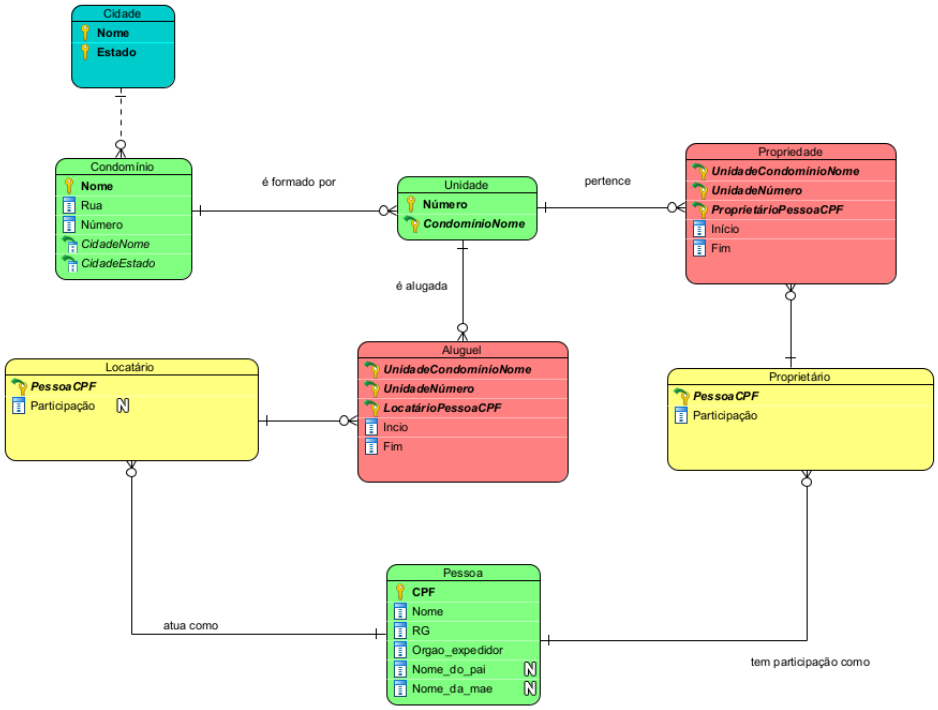


Figura 3.4: Exemplo de modelo de entidades e relacionamento usando cores de acordo com o proposto por Coad, Luca e Lefebvre (1999)

- relatórios raramente são entidades. Normalmente eles são apenas os resultados de um processo que acessa várias entidades;
- linhas de relatório geralmente são entidades;
- nomes de colunas indicam entidades ou seus atributos;
- nenhum valor calculado ou derivado é atributo ou entidade;
- substantivos em regras de negócio são normalmente entidades;
- produtos, quando não são únicos, são normalmente entidades;
- papéis, como funcionário, atendente, apostador, etc., são bons candidatos para entidade, e
- um grupo de dados que se repete em uma entrada ou saída de dados é normalmente uma entidade (ou mais).

3.5.1 Onde encontrá-las

Além de encontrá-las em entrevistas e em regras de negócio, podemos utilizar alguns documentos para encontrar entidades:

- relatórios;
- formulários de entrada de dados;
- arquivos, tanto de papel quanto no computador;

- fichas, como fichas de cadastro, de empréstimo, etc;
- pedidos, requisições e documentos do gênero;
- documentos contábeis e fiscais, como nota fiscal;
- planilhas de dados, em papel ou eletrônicas;
- listagens, registros, agendas, protocolos e outros documentos de trabalho;
- sistemas já existentes,e
- bancos de dados já existentes.

Outra forma de encontrar entidades é buscar sistemas semelhantes já resolvidos e padrões de projeto ou padrões internacionais sobre o assunto sendo tratado.

3.5.2 Descrevendo Entidades

É extremamente importante a descrição precisa de cada entidade , pois sua descrição não serve só de documentação, mas também de teste para verificar se entendemos realmente sua presença no sistema. Uma boa descrição de entidade conter os seguintes itens:

- nome, incluindo uma listagem de sinônimos e homônimos;
- definição;
- exemplos;
- atributos (veremos a seguir);
- relacionamentos (veremos a seguir);
- eventos que a utilizam (veremos no próximo capítulo);
- correlação, descrevendo outras partes da análise que se referem a ela;
- regras e exceções relacionadas a essa entidade, incluindo regras de negócio;
- outros comentários e observações, e
- uma idéia da quantidade esperada de instâncias no sistema.

Durante a definição devemos tentar responder várias perguntas, procurando deixar claro o porquê dessa entidade fazer parte do sistema. Assim devemos nos preocupar em dizer o que é essa entidade, o que faz e para que está no sistema, quando algo é ou não é uma dessas entidades, quando passa a ser ou deixa de ser, ou se é permanentemente.

Quando algum elemento passa de uma entidade para outra devemos tomar bastante cuidado para descrever as ações necessárias para tal fato.

3.6 Relacionamentos

A principal característica das entidades que compõe um sistema é se relacionarem umas com as outras. É impossível imaginar uma entidade isolada em um sistema de informação. Toda entidade deve possuir ao menos um relacionamento que a coloque em contato com as outras entidades do sistema.

Relacionamentos representam que existe alguma conexão entre entidades dentro do negócio sendo analisado [B34]. Cada relacionamento deve ser também uma regra de negócio e é utilizado em pelo menos um processo que lida com as entidades envolvidas.

Relacionamentos indicam a possibilidade de buscar um grupo de entidades a partir de outra entidade. Assim, permite encontrar os "visitantes" que "emprestaram" um "livro" específico (navegando de livro para clientes), ou descobrir que "produtos" um "cliente" "pediu" (navegando de clientes para livros). Indicam também que precisamos nos lembrar de algo que envolve, simultaneamente, duas ou mais entidades do sistema, e que essa lembrança só faz sentido quando todas as instâncias envolvidas são recuperáveis simultaneamente ou seqüencialmente.

Existem muitos relacionamentos comuns, encontrados em muitos sistemas, como "compõe" (peças compõe máquinas), "é um" (bicicleta "é um" meio de transporte), "faz" ou "gera" (cliente faz ou gera pedido), "atende" (visita atende solicitação de reparo), "usa" (cliente "usa" produto), etc.

O relacionamento "é um" é tão comum, e tão útil, que foi escolhido como um relacionamento especial em muitos métodos derivados da Modelagem Entidade e Relacionamento original. É a relação de herança, onde dizemos que uma entidade "herda" todas as características de outra entidade. A herança equivale à abstração de generalização/ especificação e será discutida mais adiante.

Outro relacionamento tão comum que mereceu um tratamento especial em muitos métodos é o relacionamento "é parte de". Esse relacionamento equivale à abstração de composição. É normal que utilizemos esse relacionamento apenas quando a parte só existe em função do todo, porém não é uma exigência muito forte.

Relacionamentos podem unir indiferentemente entidades do mesmo tipo ou entidades de tipos diferentes. Quando relaciona entidades do mesmo tipo, por exemplo, pessoas com pessoas, dizemos que é um auto-relacionamento. Ao especificar um auto-relacionamento devemos ter mais cuidado em declarar os papéis das entidades no relacionamento, além de atentar para não produzir um loop infinito no relacionamento. Para isso, algum lado do relacionamento tem que ser opcional.

Apesar de ser possível trabalhar com relacionamentos múltiplos, isto é, relacionamentos contendo mais de duas entidades, normalmente trabalhamos apenas com relacionamentos entre duas entidades. É comum transformar um relacionamento múltiplo em alguma entidade que o represente.

O mesmo acontece com o uso de atributos no relacionamento. Apesar do método original permitir, atualmente criamos uma entidade para representar esse relacionamento. Bertini, Ceri e Navathe (1992) mostram algumas operações que, aplicadas a um modelo e-r, criam diagramas diferentes que podem representar uma mesma realidade. Assim, algo que foi representado como uma entidade em um modelo pode ser representado como duas em outro, ou um relacionamento em um modelo pode ser transformado em uma entidade que se relaciona com as entidades originais (ou vice-versa) sem que haja uma

representação falsa da realidade. Pode acontecer de uma ou outra representação ser mais interessante em um contexto.

Relacionamentos podem ser condicionais ou incondicionais, isto é, uma entidade pode ser obrigada a ter um relacionamento com outra ou não. Por exemplo: um automóvel é obrigatoriamente fabricado em uma fábrica, mas nem todos os livros em uma livraria já foram vendidos. Como veremos adiante, o fato de um relacionamento ser opcional é representado pela definição da cardinalidade mínima do relacionamento, que pode ser 0 ou 1.

Também é importante notar que existem também relações que ocorrem entre relacionamentos. Dois relacionamentos podem ocorrer sempre juntos (contingentes) ou nunca ocorrer juntos (mutuamente exclusivos). Existem métodos que permitem anotar diretamente no diagrama essas características, porém são pouco utilizados.

Tudo que não puder ser anotado no diagrama deverá ser anotado em um documento associado. O principal tipo de anotações são as regras de negócio que funcionam como restrições, como "um professor só pode dar aulas para alunos da escola em que trabalha". Restrições são geralmente impossíveis de desenhar diretamente no diagrama .

Normalmente associamos restrições a ciclos no diagrama. Por exemplo, se temos que fazer pedidos de livros para uma editora, então temos um relacionamento entre livro e pedido, um entre livro e editora e um entre editora e pedido, formando um ciclo. A restrição é que "um pedido só pode conter livros da editora indicada no pedido". É possível desenhar o diagrama sem ciclos, eliminado, por exemplo, a ligação entre pedido e editora, porém aconteceriam duas coisas: primeiro teríamos que escrever uma restrição que é possivelmente mais complexa ("um pedido só pode conter livros da mesma editora"), segundo não teríamos nenhuma indicação no diagrama que o pedido é feito para a editora, exigindo uma nova regra. Finalmente, a falta do ciclo funciona também como falta de indicação que existe uma restrição, pois todo ciclo é um aviso de restrição .

3.6.1 O Relacionamento de Herança

Existem duas formas básicas de herança. Na herança exclusiva dividimos uma classe em categorias. Essa forma de herança traz poucas dificuldades na modelagem e é conhecida como separação em categorias. Uma pessoa, por exemplo, pode ser dividida em duas categorias, a dos homens e a das mulheres. Quando a divisão não é exclusiva, ou seja, quando é possível que uma instância de uma entidade específica seja classificada em duas (ou mais) de suas subclasses, temos alguns problemas que devem ser resolvidos na modelagem lógica. Por exemplo, uma pessoa pode ser aluno e professor simultaneamente em uma faculdade.

Também é possível que existam instâncias que não fazem parte de nenhum dos tipos de entidade especializados, mas fazem parte do tipo geral. Isso também exige um tratamento especial durante a modelagem lógica.

Nós dizemos então que:

- A cobertura é total, se cada elemento da classe genérica é mapeado em ao menos um elemento das subclasses.
- A cobertura é parcial, se existe ao menos um elemento da classe genérica não mapeado nas estruturas das subclasses.
- A cobertura é exclusiva, se cada elemento da superclasse é mapeado em no máximo um elemento das subclasses.
- A cobertura é sobreposta, se existe um elemento da superclasse mapeado em mais de um elemento das subclasses.

Devemos tentar obter apenas heranças totais e exclusivas, pois são mais fáceis de serem tratadas. Dado um grupo de entidades candidatas a construir um relacionamento de herança, devemos analisar se existe um atributo ou relacionamento que é aplicável a apenas um subconjunto dessas entidades, se simplificamos o modelo e se aumentamos sua compreensão. Ou seja, devemos usar a herança para aumentar a semântica do modelo sem causar excesso de informação.

3.6.2 Cardinalidade

Para bem representar um relacionamento, devemos indicar a cardinalidade desse relacionamento, isto é, quantas vezes uma instância da entidade pode se relacionar com instâncias de outras entidades.

Veja por exemplo o relacionamento "mãe-filha". Uma filha só pode ter uma mãe, mas uma mãe pode ter várias filhas. Existem três tipos básicos de relacionamentos: o 1:1, um para um, o 1:N, um para muitos, e o N:M, muitos para muitos. Nesse caso só estamos falando da cardinalidade máxima. A cardinalidade máxima indica quantas vezes uma entidade pode aparecer em um relacionamento.

No relacionamento 1:1 cada entidade só pode se relacionar com uma entidade do outro conjunto. Geralmente indica semelhança, igualdade, utilização conjunta, etc.

No relacionamento 1:N cada entidade de um conjunto pode ser relacionar com várias entidades do outro conjunto, mas as entidades do segundo conjunto só podem se relacionar com uma entidade do primeiro conjunto. Geralmente indicam relações de posse, hierarquia ou de composição.

No relacionamento N:M qualquer número de relacionamentos é válido. Podem indicar várias coisas, como eventos, contratos, acordos, ligações temporárias como empréstimos e aluguéis, etc. É normal aparecerem também quando o relacionamento é do tipo 1:1 ou 1:N em certo momento ou período (como o aluguel de uma fita de vídeo), mas se deseja manter a história de todos os relacionamentos. Quando falamos também da cardinalidade mínima usamos notação de par ordenado, (0,1):(1,N) por exemplo, onde o primeiro número do par indica a cardinalidade mínima e o segundo a máxima. A cardinalidade mínima indica uma exigência da participação de uma instância da entidade

em relacionamentos. A cardinalidade mínima 0 em ambos os lados indica a existência própria de ambos os objetos. A cardinalidade mínima 1 pode indicar a necessidade de um objeto pertencer ou ser criado por outro.

É comum evitarmos, na prática, relacionamentos onde ambos os lados exijam como cardinalidade mínima "1". O motivo é que um par de entidades só pode ser colocado na memória do sistema em uma mesma transação, não permitindo que primeiro coloquemos a instância de uma entidade na memória e depois uma instância relacionada da outra entidade. Obviamente os SGBDs permitem criar uma transação longa, mas mesmo assim isso cria uma dificuldade adicional.

Temos então os seguintes tipos de relacionamentos:

- Relacionamentos um para um.
    - O relacionamento 1x1 é menos comum em um modelo, já que a restrição que ele aplica a relação entre dois conjuntos é forte: uma instância só pode se relacionar com uma instância apenas.
    - Seus sub-tipos são:
        - ◇ (0,1):(0,1)
            - Esse relacionamento significa que uma instância do primeiro conjunto pode ou não se relacionar com uma instância do segundo conjunto, porém pode ter apenas um relacionamento. O mesmo vale do segundo conjunto para o primeiro.
            - Esse relacionamento é encontrado quando é possível formar pares entre duas entidades, mas esses pares são opcionais.
            - Um exemplo seria o caso da alocação cabines e reboques de caminhões em uma empresa de aluguel de viaturas. Cabines e reboques podem ser trocados arbitrariamente. Em certo momento, cada cabine só pode ter um reboque e vice-versa. Além disso, algumas vezes uma das partes fica guardada na garagem enquanto a outra é utilizada.
            - Outro caso que podem mostrar são auto-relacionamentos desse tipo. Uma igreja ou um templo, por exemplo, pode ter um catálogo de frequentadores e querer saber quem é casado com quem (e quem é solteiro, ou seja, não tem nenhum relacionamento).
        - ◇ (1,1) :(0,1)
            - Esse relacionamento significa que a primeira entidade obrigatoriamente tem um relacionamento, mas ele é opcional para a segunda entidade. Em ambos os casos apenas um relacionamento é permitido.
            - Esse relacionamento é encontrado quando uma entidade possui ou controla de alguma forma outra. Em alguns casos as duas entidades podem ser unidas em uma só.
            - Ele é menos comum que o relacionamento (1,1):(0,N).
            - Um exemplo seria uma distribuição de papéis de uma peça de teatro em uma companhia de atores. Cada ator só pode fazer um papel,

alguns atores podem não ter papel, mas todos os papéis têm um ator, e apenas um ator.
    - ◇ (0,1):(1,1), similar ao anterior
    - ◇ (1,1):(1,1)
        - Esse relacionamento é pouco comum, pois indica que uma entidade não pode existir sem estar relacionada com outra, e tudo isso apenas uma vez. Normalmente pode ser substituído pela unificação das duas entidades em uma só.
        - Algumas vezes é utilizado para diferenciar aspectos diferentes da mesma entidade. Por exemplo, um avião é uma entidade que tem visões comerciais, de mecânica, de operação, etc. Fica muito complicado, em um modelo ER, colocar todos os atributos, que chegam a centenas, em uma só entidade, assim podem ser criados relacionamentos (1,1):(1,1) para tratar essa modelagem.
        - Esse relacionamento não é recomendado, pois exige que ambas as entidades sejam sempre criadas juntas.
- Relacionamentos 1 para N
    - Os relacionamentos 1xN são o típico relacionamento pai-filhos, ou mão-filhas. Ele aplica uma restrição a Relação entre os conjuntos de entidades: as instâncias de um dos conjuntos só podem aparecer uma vez na relação, como um filho só pode ter um pai.
    - Outro uso comum de relacionamentos 1xN é quando queremos guardar a história de relacionamentos 1x1, por exemplo, em nosso caso anterior dos caminhões e reboques, não queremos apenas saber que caminhão está usando que reboque nesse instante, mas toda a história de uso.
    - Na prática, esses relacionamentos também são usados nos modelos mais modernos quando devemos guardar dados sobre os relacionamentos NxM, porque não temos mais o losango do modelo original de Peter Chen para guardar as entidades. Então o relacionamento NxM se transforma em dois relacionamentos 1xN para outra entidade, que registra a informação. Muitas vezes essa entidade é do tipo “evento ou intervalo”, como um contrato entre as partes.
    - Seus sub-tipos são:
        - ◇ (0,1):(0,N)
            - A primeira entidade pode ou não participar do relacionamento, mas apenas uma vez. A segunda entidade pode ou não participar do relacionamento, e ainda pode fazê-lo várias vezes.
            - Esse é um relacionamento muito comum. Normalmente significa que dois objetos que não possuem nenhum relacionamento de propriedade ou restrição de existência podem ser colocados em uma relação hierárquica.
            - Exemplo: esse tipo de relacionamento pode ser encontrado em locais onde temos um estoque de objetos que são alocados a departamentos da empresa, por exemplo, computadores. Um computador só pode

estar alocado em um departamento ou pode estar no estoque (sem alocação). Um departamento pode ter 0, 1 ou vários computadores alocados para si.

- ◇ (0,N):(0,1), similar ao anterior
- ◇ (0,1):(1,N)
  - ○ Esse relacionamento normalmente também indica uma relação hierárquica.. O primeiro objeto pode opcionalmente pertencer a essa relação, enquanto o segundo objeto obrigatoriamente pertence a relação.
  - ○ Não é muito comum, pois exige que uma instância tenha no mínimo uma "filha", mas as filhas podem existir de forma independente.
  - ○ Pode ser usado quando algo para existir deve ter ao menos uma parte, mas estas partes tem vida própria, mesmo só podendo ser usadas em um lugar.
  - ○ Isso pode ser encontrado, por exemplo, no relacionamento entre uma venda e os itens (quando únicos) vendidos. Uma loja de carros novos, por exemplo, pode em uma mesma venda negociar vários carros, mas necessariamente a venda contém um carro. Já o carro pode ter sido vendido ou não.
- ◇ (1,N): (0,1), similar ao anterior
- ◇ (1,1):(0,N)
  - ○ Indica uma "maternidade" da segunda entidade em relação à primeira, ou seja, cada instância da primeira entidade é obrigada a possuir uma "mãe", e apenas uma, que seja instância da segunda entidade.
  - ○ É um dos relacionamentos mais comuns.
  - ○ Pode ser encontrado, por exemplo, na relação entre automóveis de uma empresa e multas recebidas. Cada multa é de apenas um automóvel, mas podem existir automóveis com 0, 1 ou mais multas.
- ◇ (0,N) :(1,1), similar ao anterior
- ◇ (1,1):(1,N)
  - ○ Indica uma "maternidade" da segunda entidade em relação à primeira, ou seja, cada instância da primeira entidade é obrigada a possuir uma "mãe", e apenas uma, que seja instância da segunda entidade. Além disso, obrigatoriamente a "mãe" deve possuir uma filha.
  - ○ Esse relacionamento apresenta o inconveniente de exigir criar uma entidade "filha" para criar a entidade "mãe".
  - ○ Pode ser encontrado, por exemplo, em um cadastro de pessoas jurídicas, que devem possuir um endereço, mas podem possuir mais de um.
  - ○ Também é encontrado na modelagem normatizada de objetos que contém listas que obrigatoriamente possuem um item, como uma nota fiscal.
- ◇ (1,N):(1,1) , similar ao anterior

- Relacionamentos N para M

- Os relacionamentos NxM são a forma mais geral, que menos restrições oferecem a relação entre os dois conjuntos de entidades envolvidos, porém não pode ser implementado diretamente em um SGDB, exigindo a criação de uma tabela onde são anotados os pares de instâncias de entidades envolvidos.
- Muitas ferramentas não desenham esse tipo de relacionamento, por não ser possível sua implementação direta nos SGDBs.
- Seus sub-tipos são:
  - (0,N):(0,N)
    - Esse relacionamento é muito comum. Representa a forma mais geral de relacionamento, opcional e com todas as possibilidades para ambos os lados.
    - Pode ser encontrado, por exemplo, na relação entre alunos e cursos oferecidos em um semestre em uma universidade. Alguns cursos não recebem inscrição, alguns alunos não fazem inscrição em nenhum curso.
  - (0,N):(1,N)
    - Semelhante ao (0,N):(0,N). Também muito comum, porém agora exigimos que haja pelo menos um relacionamento na segunda entidade.
    - Pode ser encontrado, por exemplo, na relação entre músicas e CDs onde estão gravadas, para controle de uma discoteca. Uma mesma música pode estar em vários CDs, mas não é possível registrar um CD sem músicas (deve existir pelo menos uma). Porém uma música pode nunca ter sido gravada.
  - (1,N):(0,N), similar ao anterior
  - (1,N):(1,N)
    - Aqui temos um relacionamento múltiplo que deve existir pelo menos uma vez.
    - Um exemplo é o relacionamento entre salas de uma empresa e móveis colocados nessa sala.
    - Essa representação muitas vezes é verdadeira, mas é evitada, sendo trocada pelo relacionamento (0,N):(1,N), pois exige que ambas as entidades, quando estão sendo criadas, sejam sempre criadas juntas, ou que existam algumas entidades na base como "semente".

3.6.3 Descrevendo Relacionamentos

Dependendo da notação, relacionamentos podem ser descritos por linhas ligando duas entidades ou por um losango ligado por linhas às entidades. Em ambos os casos é possível anotar os relacionamentos com nomes e com a sua cardinalidade (ver exemplos mais a frente).

O nome escolhido para o relacionamento pode estar na voz ativa (mãe gera filho) ou na voz passiva (filho é gerado por mãe). Algumas notações permitem que se usem os dois

nomes (um por cima e um por baixo da linha de relacionamento). Geralmente se usa o nome que permite a leitura do relacionamento da esquerda para a direta na parte de cima da linha (ou se dá preferência a esse nome quando apenas um pode ser utilizado).

Relacionamentos também devem ser descritos e comentados, sendo importante responder qual sua função no sistema, o que eles representam, como e quando são estabelecidos ou destruídos.

3.7 Atributos

Todo atributo descreve de alguma forma a instância da entidade. Alguns atributos são especiais e definem a entidade, mesmo que não de forma unívoca. Esses são os atributos nominativos. Outros atributos permitem definir outro objeto que não é o sendo tratado, são ou atributos referenciais. Um exemplo de atributo referencial é "fábrica" para "automóvel", referenciando a fábrica onde foi construído. É uma opção do analista criar ou não entidades que permitem a substituição de um atributo referencial por um relacionamento .

3.7.1 Descrevendo Atributos

Devemos definir as seguintes características:

- Nome
- Descrição
- Domínio (valores válidos, como inteiro, real, string ou uma lista de valores, ou ainda tipos criados pelo projetista).
- Tipos de nulos aceitos Exemplo

Na descrição devemos nos preocupar em explicar qual a finalidade do atributo, como são atribuídos os valores, o que significa cada valor, quem define a escolha do valor, quando, por que e por quem o valor é atribuído ou alterado, etc. Atributos são atualmente denotados no mesmo retângulo da entidade, como mostrado nos exemplos a seguir.

3.7.2 Atributos Identificadores (Chaves Candidatas e Chaves Primárias)

Alguns atributos têm o poder de distinguir as ocorrências das entidades, isto é, servem para identificar univocamente uma instância de entidade à instância do mundo real . Definido o valor desse atributo, os outros valores são dependentes e não podem ser escolhidos, mas sim devem possuir um valor exato seguindo a realidade.

Um atributo identificador típico em sistemas financeiros é o CPF de uma pessoa física ou o CNPJ de uma pessoa jurídica. Definido o CNPJ, a empresa, e todos os seus dados,

estão univocamente definidos (nome fantasia, endereço, etc.) no mundo real, e assim deve seguir o sistema que estamos construindo.

Muitas vezes precisamos de mais de um atributo identificador para realmente identificar uma instância. Dizemos então que a chave primária é composta. Se usarmos apenas um atributo como identificador, então dizemos que a chave primária é simples.

3.7.3 Relacionamentos Identificadores

Algumas instâncias são identificadas também, ou até mesmo unicamente, por seus relacionamentos. Uma forma de denotar isso é utilizar uma linha mais grossa no relacionamento ou algum símbolo específico. Alguns autores chamam as entidades que são identificadas por seu relacionamento com outras entidades de "entidades fracas" ou "entidades dependentes". Atualmente esses nomes são considerados derrogatórios para entidades que podem ser muito importantes em um modelo. Os alunos também, muitas vezes, tendem a achar que não devemos modelar "entidades fracas", conclusão que está absolutamente errada.

Chaves Estrangeiras

No modelo conceitual não existe o conceito de chave estrangeira, que é uma característica do modelo relacional. Uma chave estrangeira é uma chave de outra tabela que usamos em uma tabela para indicar o relacionamento. Porém, é comum que as ferramentas de modelagem copiem as chaves estrangeiras automaticamente . Em benefício da prática atual, e em detrimento da pureza teórica, mostramos a seguir algumas possibilidades da notação de relacionamento.

3.8 Descrição Gráfica do Modelo

Várias são as notações existentes para o modelo de entidade e relacionamento. Usaremos nesse texto a notação de Martin, também conhecida como Information Engineering, fornecida pelo software Erwin.

Nessa notação não temos um símbolo para relacionamentos, apenas um retângulo para entidades. Os relacionamentos são indicados por linhas. Uma linha cheia indica um relacionamento identificador. Uma linha tracejada indica um relacionamento não identificador. Por isso, não podemos usar relacionamentos com atributos, necessitando de uma nova entidade nesse caso. Também não podemos criar relacionamentos múltiplos, necessitando de criar entidades para representá-los.

Apesar de parecer que temos um modelo menos poderoso, temos na verdade apenas uma sintaxe mais simples, com o mesmo poder de modelagem. Algumas decisões também ficam tomadas automaticamente também. Por exemplo, não precisamos decidir se um

"objeto com atributo" é um relacionamento ou uma entidade, pois relacionamentos não têm atributos em nosso modelo. Acreditamos que a modelagem segundo as regras do IDEF1X ou da Engenharia de Informação possibilita encontrar mais facilmente um modelo essencial do sistema que as regras tradicionais de Chen ou ainda extensões as mesmas.

A cardinalidade é indicada por três símbolos usados na ponta da linha que indica o relacionamento: uma linha indica 1, um círculo indica 0 (zero), e um "pé-de-galinha" indica n. Dessa forma podemos anotar o mínimo e o máximo da cardinalidade usando dois símbolos em cada ponta.

O nome do relacionamento é colocado acima (à esquerda) da linha que o indica, sendo o nome do relacionamento inverso colocado abaixo (à direita). Normalmente se lê a notação partindo de uma entidade, lendo o nome do relacionamento, lendo a cardinalidade da ponta oposta e finalmente o nome da segunda entidade.

Nessa notação os atributos podem ser colocados dentro da caixa que representa as entidades, como apresentado na próxima seção.

3.9 Exemplos de notação da Engenharia de Informação

Vamos descrever um modelo na notação da Engenharia da Informação, também conhecida no Brasil como "pés de galinha".

Apresentaremos o modelo de uma locadora de vídeo. A locadora trabalha com fitas de vídeo. Cada fita de vídeo contém um filme, porém cada fita deve ser identificada unicamente, pois elas podem ser dubladas ou legendadas. As fitas são emprestadas para clientes em um dia e hora específico. Um cliente pode ficar com várias fitas, ou nenhuma. Uma fita pode estar com apenas um cliente, ou estar na loja e não estar com cliente nenhum. É importante saber para quem cada fita específica foi emprestada, para auditar clientes que estragam fitas, por isso todas as fitas são numeradas com um código único. Os filmes são dirigidos por diretores e contém atores.

Observamos que o cliente é um papel assumido por uma pessoa, a fita de vídeo é um objeto físico, existente, o filme é uma obra de arte que está representada na fita (um conceito), diretor e ator são também papéis assumidos por pessoas dentro da idéia de filme e que um aluguel é um contrato entre o cliente e a locadora.

Atenção para outro detalhe: não existe a entidade locadora, pois este sistema é destinado a uma só locadora. Seria uma entidade única, que claramente é uma constante do sistema. A presença de entidades desse tipo é um erro comum nos modelos feitos por principiantes. Porém, se tivéssemos um software para uma rede de locadoras, seria interessante guardar em que locadora está cada fita, o que exigiria essa entidade.

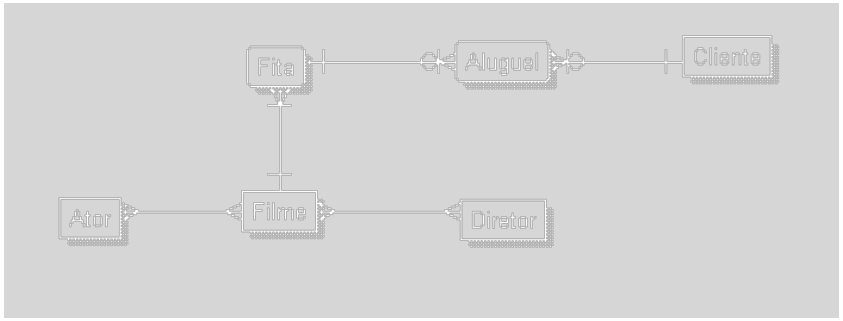


Figura 3.5: Modelo ER só com entidades e relacionamentos, notação da Engenharia da Informação

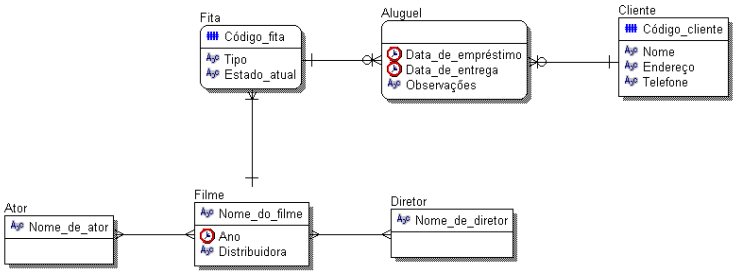


Figura 3.6: Modelo ER com atributos, notação Engenharia da Informação

3.10 Exercícios

Exercício 3.1: Escreva um modelo de entidades e relacionamentos para um aplicativo de celular que sirva como relógio e despertador, permitindo vários alarmes.

Exercício 3.2: Vá para o site `http://jogodeanalisedesistemas.xexeo.net/` e visite a Livraria Resolve. A partir da sua visita faça um modelo de entidades e relacionamentos para o sistema proposto.

4 Bases Exemplo

Neste capítulo vamos mostrar dois bancos de dados de exemplo, descritos com o Modelo de Entidades e Relacionamentos. A primeira é a base Sakila e a segunda a NorthWind.

4.1 O Bando de Dados Sakila

O modelo de dados da Figura 4.1, escrito na notação de Engenharia de Informação, descreve o banco de dados conhecido como **Sakila**(Oracle Corporation, 2019c). Esse banco é disponibilizado pela Oracle como um banco de dados exemplo de MySQL, licenciado com uma licença New BSD(Oracle Corporation, 2019a).

O modelo descreve uma empresa que possui várias lojas que alugam filmes, sem discutir o formato dos mesmos (DVDs?). Apesar de ser um tipo de negócio em decadência, ainda serve para o nosso propósito.

A informação central do modelo, que aparece na Figura 4.2 está na tabela que registra os aluguéis (*rental*) e na que registra os pagamentos (*payment*).

Essas tabelas se referem a basicamente três áreas: os clientes (*customer*), que alugam (*rental*) itens de estoque (*inventory*), sendo atendidos por um funcionário (*staff*). Esse itens são cópias de filmes (*film*) e os funcionários trabalham em lojas (*store*).

Os pagamentos são feitos pelos clientes aos funcionários para pagar um aluguel.

Clientes, funcionários e lojas possuem endereço (*adrress*), que é uma tabela única para todos. Endereços estão em cidades (*city*) que estão, por sua vez, em países (*country*).

Todos os funcionários estão alocados em alguma loja, e alguns funcionários são gerentes de uma ou mais lojas.

Os filmes são classificados (*film_category*) em categorias (*category*), e também organizados (*film_actor*) pelos atores (*actor*) que trabalham nele. Todo filme também possui

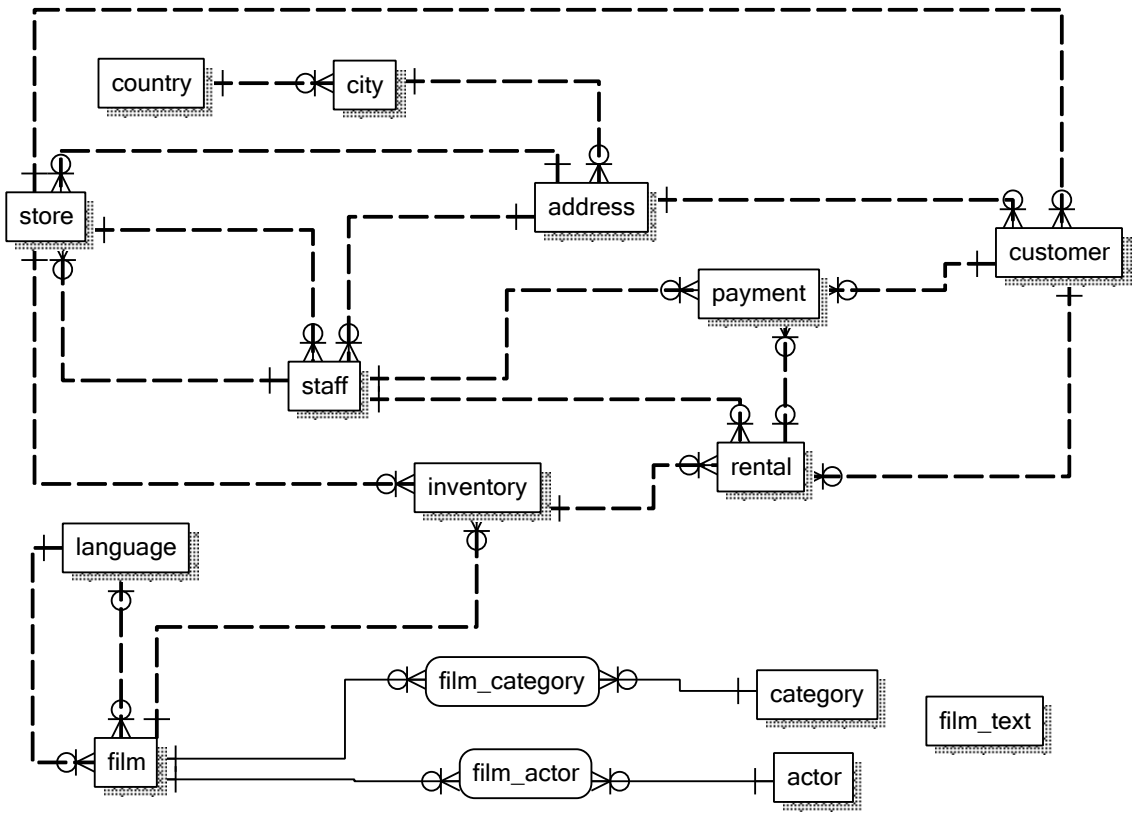


Figura 4.1: O modelo de dados da base sakila

uma pela linguagem (*language*) original e linguagem disponível o que implica em dois relacionamentos distintos.

A Figura 4.2 apresenta o modelo mais detalhado, com os nomes dos campos, onde as chaves estrangeiras permitem entender como os relacionamentos foram implementados.

Com os nomes dos campos, e mesmo sem os tipos, podemos ver que a principal informação de faturamento do negócio está na tabela *payment* e se refere ao valor (*amount*) do aluguel.

É importante ver que nessa base, toda criação ou atualização de uma linha de tabela é anotada no campo *last_updade*, que aparece em todas as tabelas.

Também existe uma tabela extra que repete alguns dados dos filmes (*film_text*).

A base Sakila contém apenas duas lojas e dois funcionários. Também possui 599 clientes, 16005 aluguéis, 16086 pagamentos, 1000 filmes, 4581 itens de inventário, 10 categorias, 603 endereços e 200 atores cadastrados.

Finalmente a Figura 4.3 mostra o modelo de dados com seus campos e os tipos dos campos. Vemos que os valores financeiros são guardados com o tipo *decimal*. As datas

usam o tipo *datetime*, mas algumas informações, como *last_update*, que aparece em todas as tabelas usam o tipo *timestamp*.

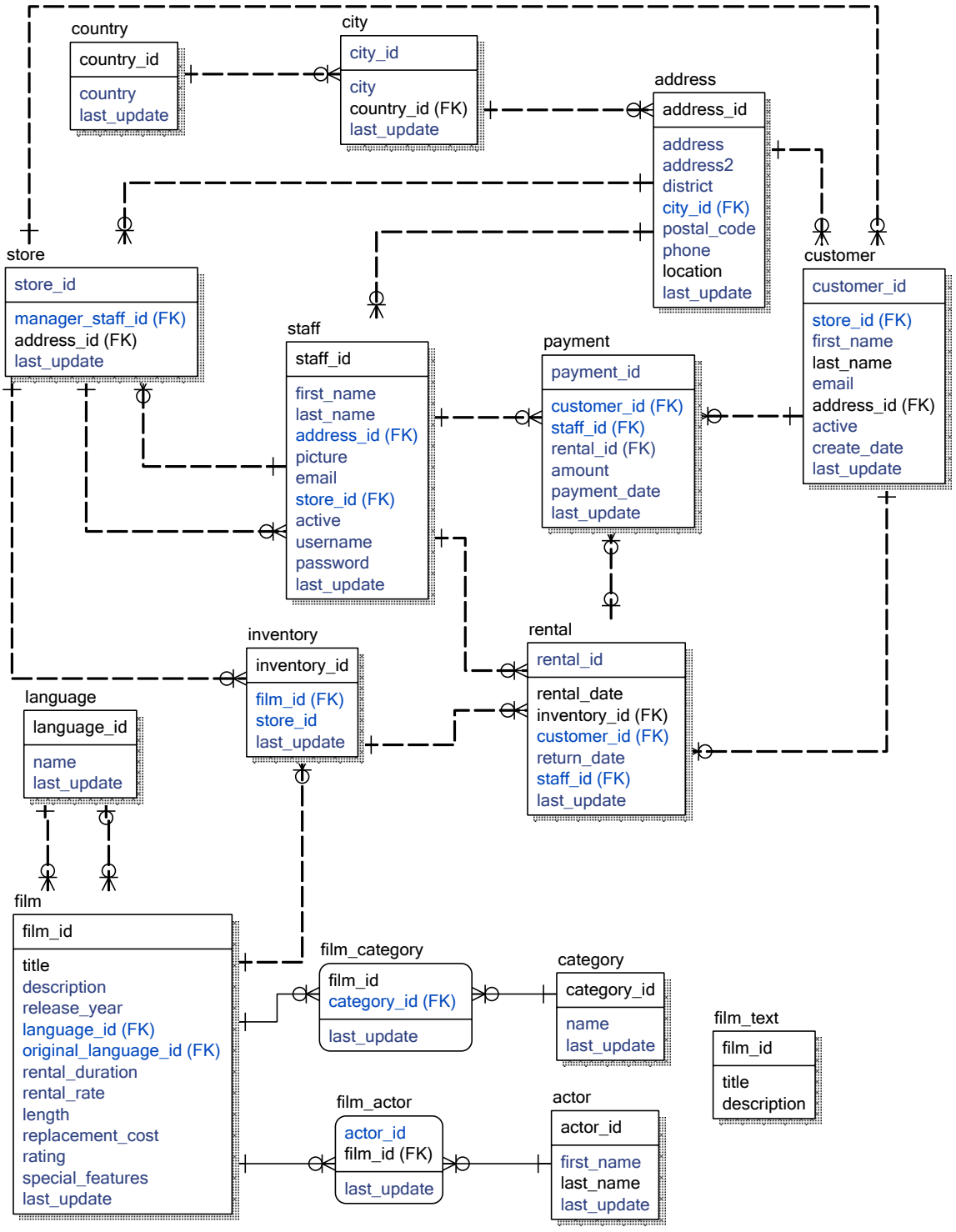


Figura 4.2: O modelo de dados da base Sakila com os nomes de campo

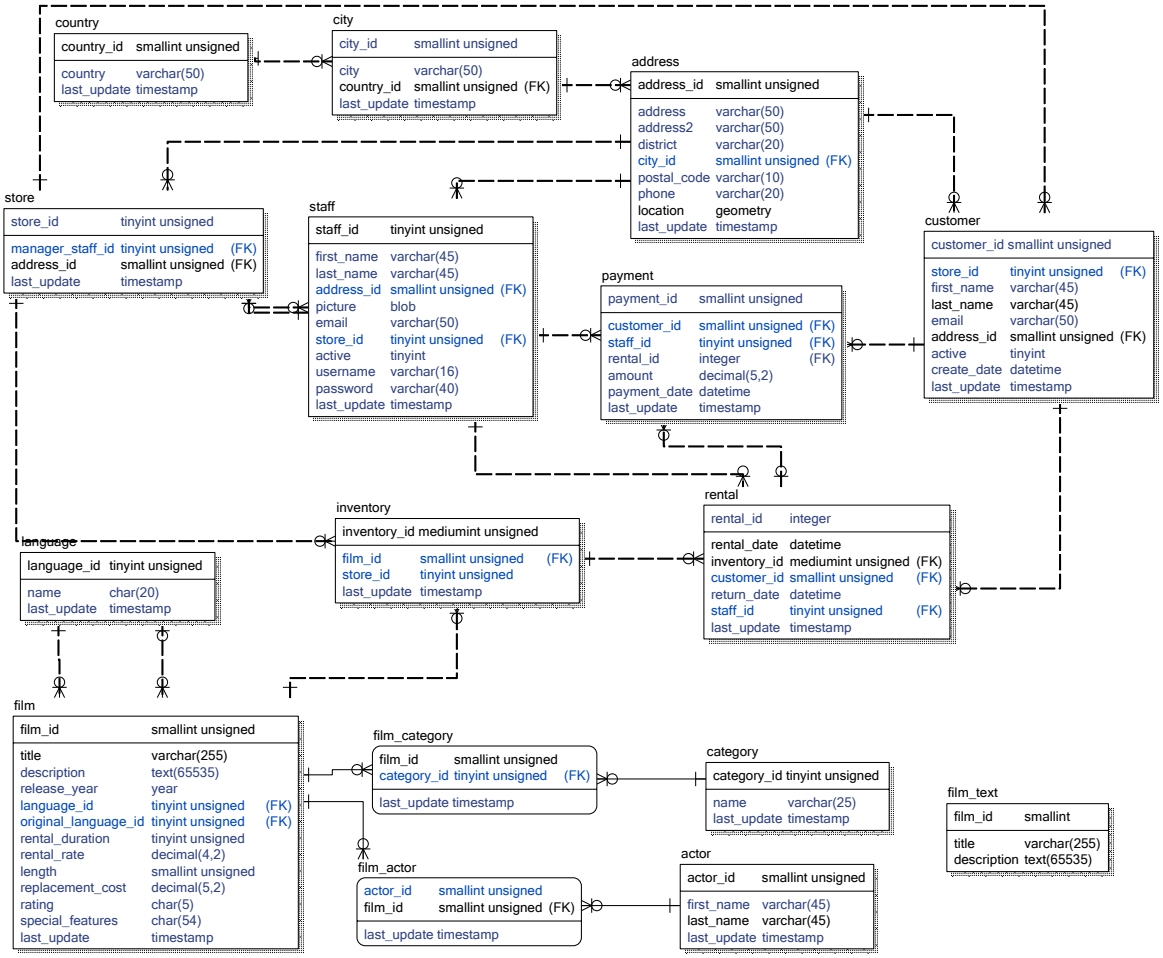


Figura 4.3: O modelo de dados da base Sakila com nomes de campo e seus tipos

4.2 O Banco de Dados Northwind

O banco de dados Northwind é um banco de dados fornecido pela Microsoft e que simula informações de um ERP de uma empresadir . A versão usada é convertida para MySQL por Scott (2016), e ainda existe em outros formatos, como PostGres. É um banco ainda pequeno e fácil de entender, pois tem poucas tabelas.

A Figura 4.4 mostra o modelo de dados da base Northwind só com suas entidades, na notação da Engenharia da Informação.

Nesse modelo clientes (*customers*) fazen pedidos (*orders*). Os pedidos tem um estado (*order_status*), são enviados por um transportador (*shippers*), tem um estado do imposto (*orders_tax_status*) e tem uma fatura (*invoices*) correspondente.

Os pedidos são compostos de itens de pedidos, que descrevem (*order_details*) de produtos (*products*), que possuem um estado próprio (*order_details_status*).

Os produtos estão em um estoque, do qual entram e saem por meio de transações (*inventory_transactions*) de certos tipos (*inventory_transaction_types*). Eles são comprados por meio de ordens de compras (*purchase_orders*), que também estão descritas por seus detalhes (*purchase_orders_details*) e tem um estado (*purchase_order_status*). Ordens de compra são feitas para fornecedores (*suppliers*).

Quem faz pedidos e ordens de compra são os funcionários (*employees*), que possuem privilégios (*employee_privileges*) de certos tipos (*privileges*).

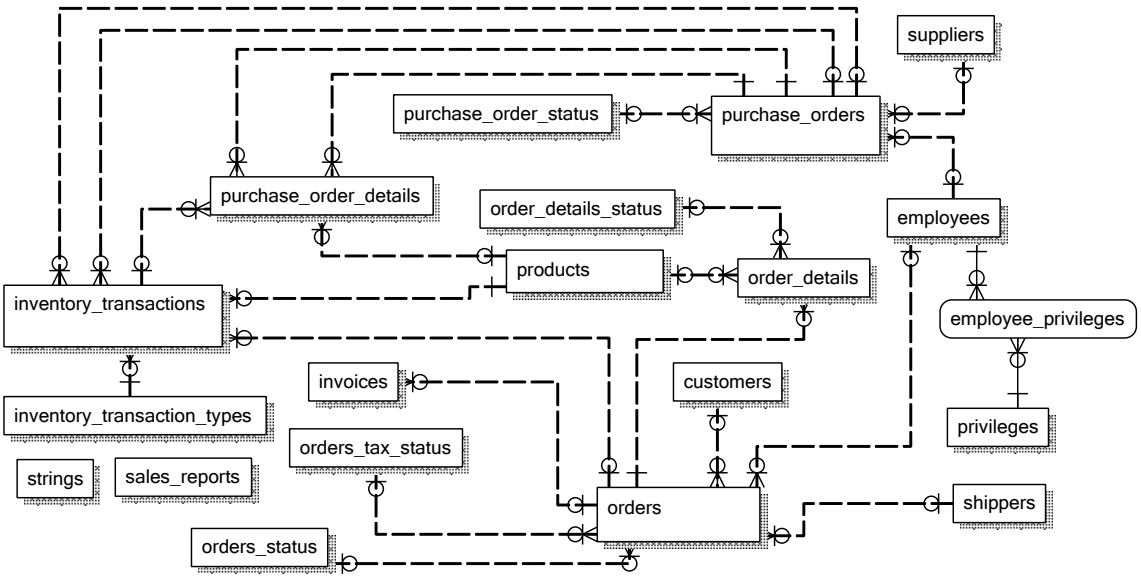


Figura 4.4: O modelo de dados da base Northwind só com as entidades

A Figura 4.5 já mostra o modelo da mesma base com atributos.

A base Northwind oferece 45 produtos, 28 ordens de compra, 9 empregados, 48 pedidos e 29 clientes. Sua tabela mais longa é a de transações do estoque, como 102 linhas.

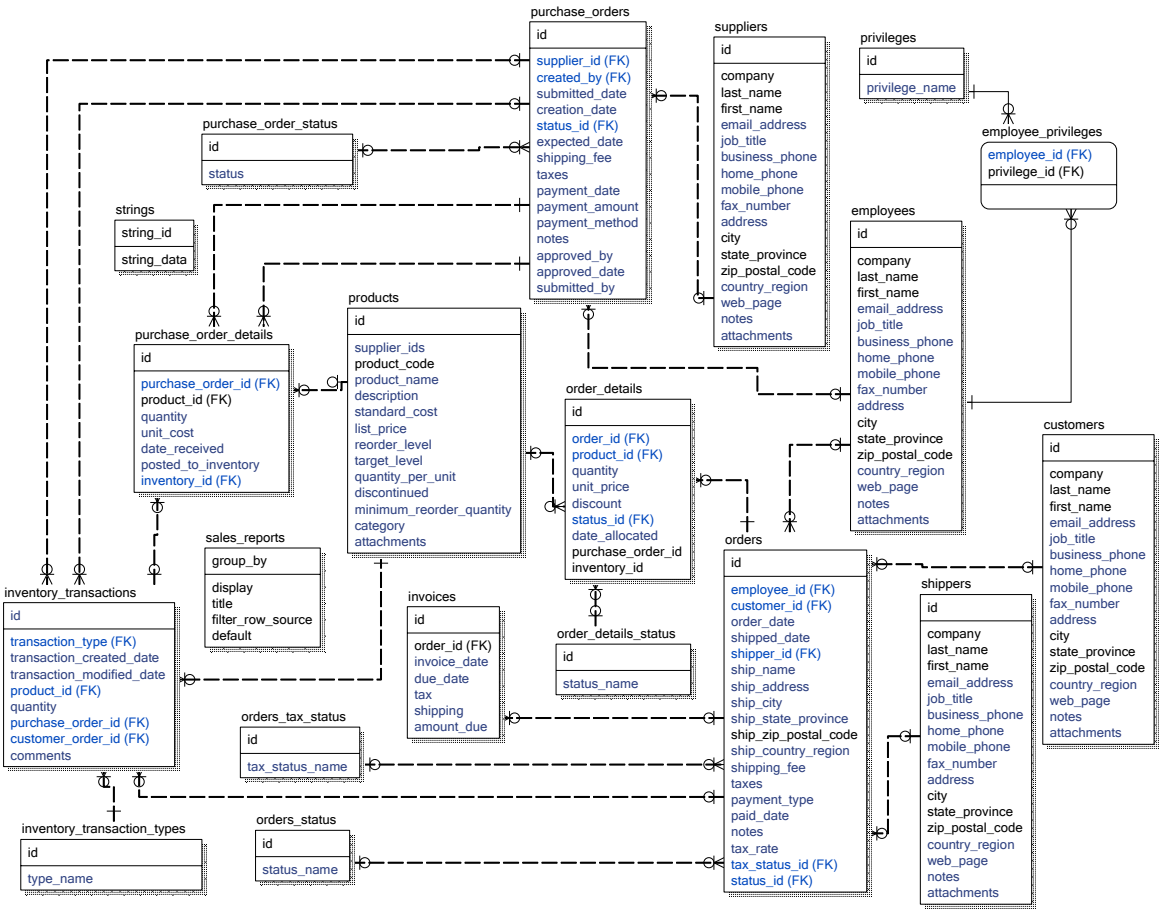


Figura 4.5: O modelo de dados da base Northwind com nomes de campo

Já a Figura 4.6 mostra não só os atributos como os tipos dos atributos. Aqui é importante notar que os valores financeiro e quantidades estão normalmente descritos como *decimal*.

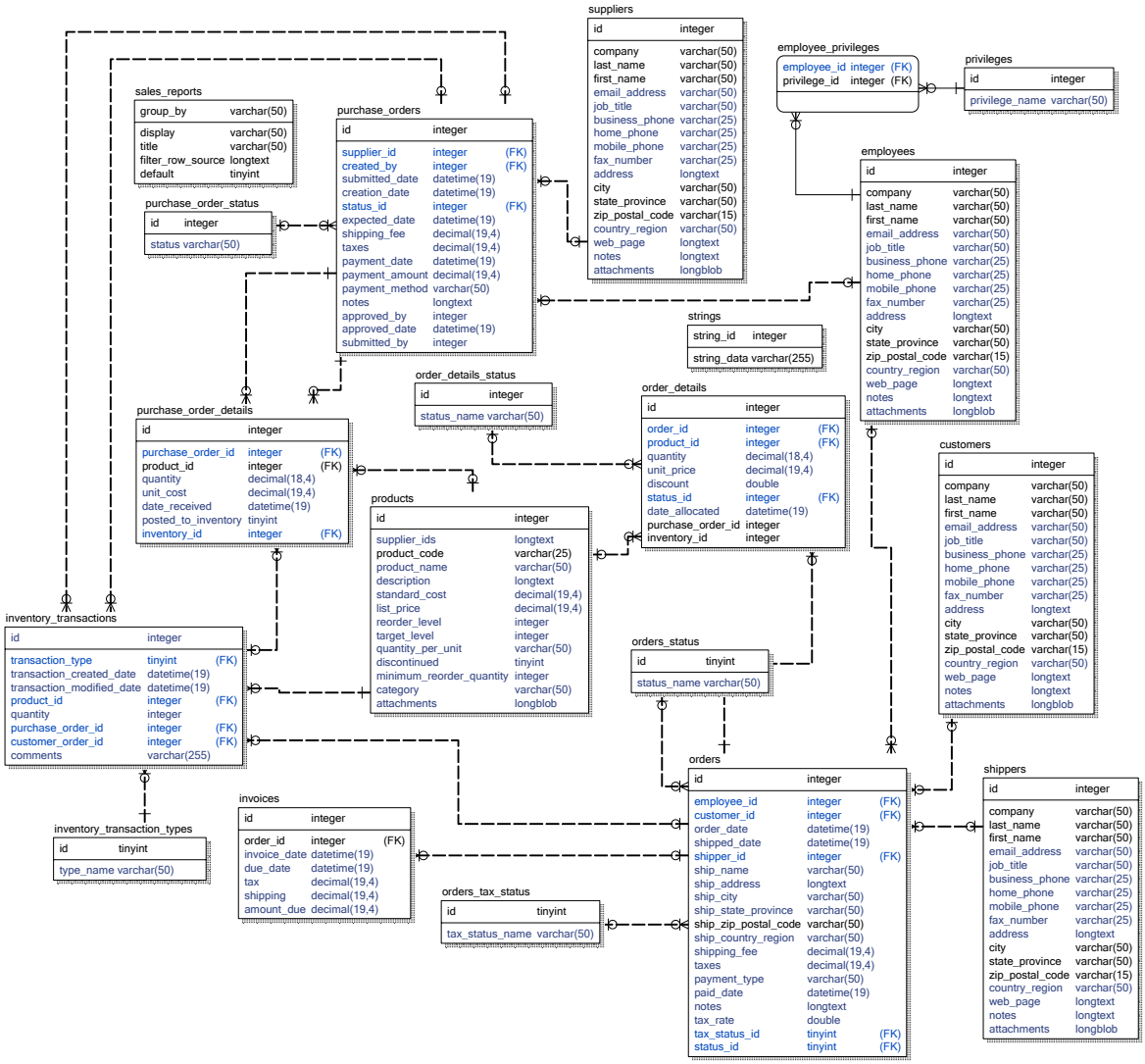


Figura 4.6: O modelo de dados da base Northwind com nomes de campo e seus tipos

4.3 A Cadeia de Valor da Northwind

O processo Pedir Produto, na Figura 4.7, possui 4 passos. Primeiro é necessário que alguém submeta uma necessidade de compra. A seguir a ordem de compra é criada. Criada, a ordem de compra fica esperando a aprovação pelo Vice-Presidente de Vendas, Depois dessa aprovação ela pode ser enviada para o fornecedor.

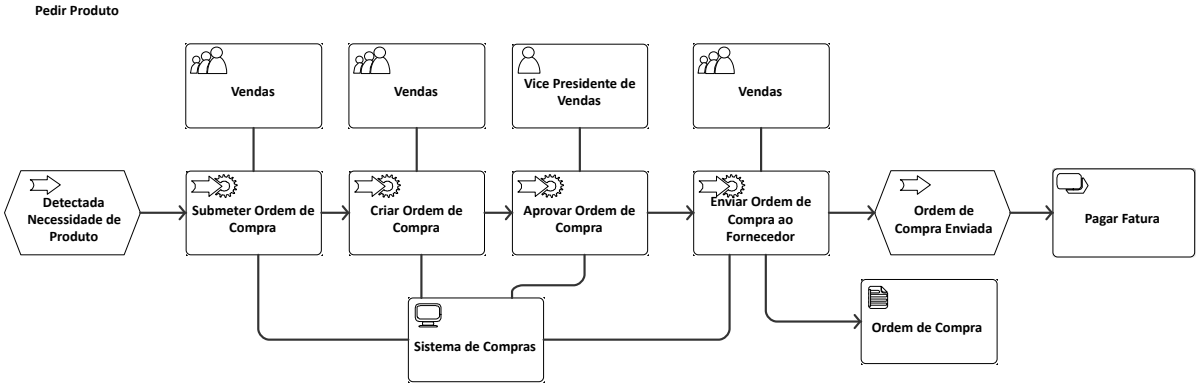


Figura 4.7: O processo Pedir Produto da Northwind

4.4 O Banco de Dados Adventure Works

O banco de dados Adventure Works 2014 (OLTP version) é "Uma base de dados exemplo para o Microsoft SQL Server, que substituiu a base exemplo Northwind, que foi lançada anteriormente. O banco de dados é sobre uma fabricante multinacional de bicicletas, fictícia, chamada *Adventure Works Cycles*.(Motl, 2019)

Esse banco contém bastante entradas e serve para alguns testes mais práticos e capacidade de Data Warehousing.

5

Bancos de Dados Relacionais e SQL

O objetivo deste capítulo é servir de uma pequena introdução ou revisão aos conceitos básicos de bancos de dados relacionais e SQL, que permita o leitor entender o restante do material apresentado no texto. O leitor encontrará cobertura bem mais completa em livros dedicados aos assuntos, como (Elmasri e Navathe, 2016).

Um **Banco de Dados Relacional** é um repositório de dados composto unicamente por tabelas, destinado a registrar as informações que representam o estado de uma sistemas de informação.

Cada tabela é formada por linhas e colunas. As linhas representam fatos de um mesmo tipo sobre o qual se deseja guardar informação, enquanto as colunas representam atributos a serem registrados sobre esses fatos, ou seja, as informações específicas que são guardadas sobre eleElmasri e Navathe (2016). Esses fatos se referem a objetos do mundo real, eventos, momentos, contratos, etc., e são diretamente ligados as instâncias de entidades ou relacionamentos.

Uma célula $a_{i,j}$ da tabela A indica o valor do atributo j para um fato i. Essa maneira de representar o mundo é conhecida como **Modelo Relacional**.

A Figura 5.1 mostra informalmente uma pequena parte de um banco de dados relacional com três tabelas, criado a partir do modelo ER da 5.2, tratando de avaliações de especialistas sobre o custo de recuperação de terrenos com problemas de contaminação. Cada especialista é descrito por seu nome, em que empresa trabalha e qual sua especialidade. Cada local é descrito por um código, o tipo de terreno e o desenvolvimento da região. Uma terceira tabela indica qual o custo de recuperação de um local de acordo com um especialista. Pela forma como foi construída, cada especialista só pode dar uma avaliação por local, porém todas as combinações de local e especialistas são possíveis, porém apenas algumas são verdadeiras e são registradas na tabela.

Figura 5.1: Exemplo de um banco de dados relacional simples

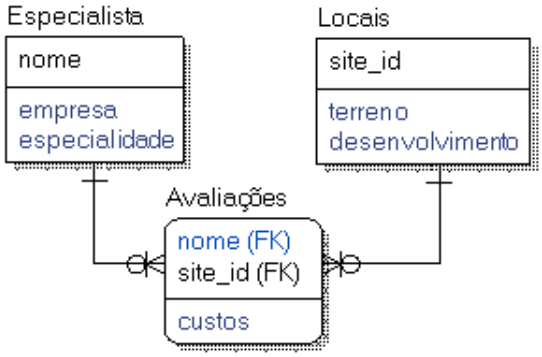


Figura 5.2: Diagrama de entidades e eelacionamentos relativo ao banco de dados da Figura 5.1

Essa é uma forma de explicar o que é um Banco de Dados Relacional em linguagem corrente. Os termos tabelas, linhas e colunas são muito fácil de entender e levam ao usuário a pensar em coisas com que tem o hábito de trabalhar.

Bancos de dados relacionais se assemelham de muita formas com o modelo de entidades e relacionamentos, sendo que a transformação de um MER em um MR é razoavelmente direta. Em linhas gerais, relacionamentos $N \times M$ devem ser eliminados, sendo transformados em novas entidades e relacionamentos $1 \times N$, e então todas as entidades são convertidas em tabelas, enquanto os relacionamentos são indicados por um campo adicional em uma das tabelas. Heuser (2001) apresenta uma descrição de alta qualidade de como fazer essa transformação.

5.1 Definição Formal de um BD Relacional

É interessante também possuir uma noção da definição formal de um banco de dados relacional, como a apresentada por Elmasri e Navathe (2016).

O Modelo Relacional usa o conceito de relação matemática como base de sua construção. Ele foi proposto por Codd (1970).

Um **domínio** D é um conjunto de valores atômicos. Normalmente um domínio é especificado de acordo com um **tipo de dados**, que define todos os valores possíveis dos valores desse domínio.

Exemplos de domínios são: datas, números de telefone, nomes de pessoa, quantidade em dinheiro, etc.

Também é comum que seja especificado um formato para cada domínio. Por exemplo, um formato válido para número de telefone pode ser "+999 999 99999-9999", indicando três números para o código do país, três para o código de cidade e nove para o número do telefone. Já um domínio relacionado a dinheiro pode ter seu valor máximo limitado e uma indicação de como será representado.

Sendo assim, para definir um domínio são necessários:

- um nome,
- um tipo de dados,
- um formato e
- outras informações adicionais que auxiliem a interpretação do. domínio

A definição do domínio é arbitrária, mas segue a lógica de um conceito desejado em um contexto específico. A Tabela 5.1 apresenta a definição de dois domínios diferentes para o mesmo conceito "sexo", para serem usados em contextos diferentes.

No primeiro caso, o sexo é definido de acordo com o padrão ISO/IEC 5218 (ISO/IEC, 2004). No segundo, segundo as necessidades de um consultório médico.

Tabela 5.1: Dois tipos de domínio para o mesmo conceito

	ISO/IEC 5218	Médico
Nome	Sexo	Sexo
Tipo de Dados	{0,1,2,9}	F ou M
Formato	o número em Unicode	a letra em Unicode
Significado	0 - não informado	F - Feminino
	1 - masculino	M - masculino
	2 - feminino	
	9 - não se aplica	

Um **esquema de relação** é denotado por

$$R(A_1,A_2,...,A_n) \tag{5.1}$$

e contém um nome de relação R e uma lista finita de **atributos** $A_1,A_2,...,A_n$ onde cada atributo A_i é um papel que algum domínio assume na relação R.

D é chamado o domínio de A_i, $D = dom(A_i)$. O **grau da relação** é o número de atributos n. O esquema da relação descreve uma **relação** chamada R.

Os esquemas de relação da Figura 5.1são:

- Especialista(nome, empresa, especialidade)
- Local(site_id, terreno, desenvolvimento)
- Avaliação(nome, site_id, custos).

Uma **relação** do esquema de relação $R(A_1,A_2,...,A_n)$, denotada por $r(R)$ é um conjunto de n-tuplas $r = \{t_1,t_2,...,t_k\}$, onde cada tupla é uma lista ordenada de valores $t =< v_1,v_2,...,v_n >$ onde cada v_j, $1 \leq j \leq n \Rightarrow v_j \in dom(A_j) \vee v_j =$ nulo.

Ou seja, na definição formal, um esquema de relação explica como é organizada a relação, que é a tabela, já preenchida com os dados, da definição informal.

Para um mesmo esquema de relação podem existir várias, possivelmente infinitas, relações possíveis. Em um certo instante do tempo, porém, provavelmente apenas uma relação representa corretamente a informação necessária para um sistema de informação.

Como uma relação não é ordenada, então as tuplas de uma relação não são ordenadas.

É importante notar que dentro do Modelo Relacional só existem tabelas. Assim, para consultar a base de dados são feitas operações com tabelas que geram outras tabelas, com o resultado desejado da consulta imaginada. Dentro da teoria foram desenvolvidos tanto uma álgebra quanto um cálculo relacional, que definem operações que sempre geram tabelas como resultadoElmasri e Navathe (2016). A linguagem SQL é uma tentativa de implementar a álgebra relacional por meio de uma linguagem declarativa.

5.2 Sistemas Gerenciadores de Bancos de Dados

Bancos de dados relacionais se tornaram ubíquos em sistemas computacionais porque criam uma camada de abstração entre vários sistemas e um conjunto comum de dados para a organização.

Os dados são armazenados, gerenciados e acessados por meio de **Sistemas Gerenciadores de Banco de Dados**, **SGDB**, que permitem o controle e acesso aos dados por meio de uma linguagem padronizada conhecida como SQL.

Assim, o que a grande maioria das aplicações faz é garantir a visualização adequada e a lógica correta de negócio e consultar e guardar os dados em um SGDB Relacional, por meio de comandos nessa linguagem.

Existem muitos SGDB no mercado, boa parte deles seguindo o modelo relacional. Entre os mais conhecidos e usados estão: MySQL, PostgreSQL, Microsoft SQL Server, e Oracle Database. Os dois primeiros são open-source e possuem licenças para uso gratuito. O Microsoft SQL Server e o Oracle são proprietários, mas tem uma licenças gratuitas para desenvolvimento e uma versão Express para pequenas aplicações, também grátis[1].

5.3 SQL

SQL[2], sigla que significa **Structured Query Language**, é uma linguagem de consulta e manipulação de banco de dados onde o programador explicita o que ele deseja e não como ele deseja que a operação seja feita. Dessa forma, SQL é uma linguagem declarativa e padronizada, apesar de cada vendedor possuir algumas extensões.

A linguagem pode ser dividida em Várias partes. A maioria dos autores registra duas partes principais: **Data Manipulation Language** (**DML**), **Data Definition Language**(**DDL**).

Outros autores ou manuais de sistemas específicos apresentam ainda variações com os seguintes subconjuntos: **Data Query Language** (**DQL**), **Data Transaction Language** (**DTL**) e **Data Control Language**(**DCL**). Este texto usa a divisão mais detalhada para melhor explicar o espectro total de SQL.

A seguir será feita uma breve introdução a algumas instruções da linguagem SQL, com mais foco na instrução SELECT. Para maiores informações devem ser procurados livros textos, como (Elmasri e Navathe, 2016) e sites e manuais das ferramentas específicas. Nesta introdução é usada a linguagem do sistema MySQL(Oracle Corporation, 2019b), devido a seu largo uso na criação de web sites e na academia.

[1]Informações válidads em novembro de 2019

[2]Deve ser pronunciada S-Q-L,letra a letra, em português ou como a palavra *sequel* em inglês.

5.3.1 Data Definition Language

A função da **Data Definition Languagem** é permitir definir e alterar a estrutura do banco de dados, isto é, criar, alterar e modificar tabelas e outros objetos do banco de dados. Deste modo, a DDL trata dos esquemas de relação, ou seja, das estruturas das tabelas.

Esse subconjunto de SQL é formado pelos comandos CREATE, ALTER, DROP, RENAME, TRUNCATE.

Neste texto serão tratados as instruções CREATE, DROP que criam e apagam tabela e outros elementos de um banco de dados.

A instrução CREATE TABLE

A instrução **CREATE TABLE** é usada para criar uma tabela. Sua sintaxe permite definir o nome da tabela, seus campos, os tipos de dados dos campos, chaves e índices.

O formato mais simples da instrução é o seguinte:

```
CREATE TABLE nome_da_tabela (
    nome_da_coluna_1 tipo_de_dados PRIMARY KEY,
    nome_da_coluna_2 tipo_de_dados,
    nome_da_coluna_3 tipo_de_dados,
   ...
);
```

Para criar as tabelas da Figura 5.1 seriam necessários os comandos:

```
CREATE TABLE Especialista
(
        nome                CHAR(255) PRIMARY KEY,
        empresa             CHAR(255),
        especialidade       CHAR(255)
);

CREATE TABLE Locais
(
        site_id             VARCHAR(20) PRIMARY KEY,
        terreno             VARCHAR(20),
        desenvolvimento     VARCHAR(20)
);

CREATE TABLE Avaliacoes
(
        nome                CHAR(255) PRIMARY KEY,
```

```
        site_id              VARCHAR(20) PRIMARY KEY,
        custos               DECIMAL(19,4),
);
```

A instrução CREATE TABLE fornece variações poderosas que normalmente podem ser encontradas no manual de cada ferramenta. Além disso, para otimizar a base seria necessário criar restrições adicionais e índices. Sua sintaxe descrita no manual do MySQL(Oracle Corporation, 2019b) contém 147 linhas e ainda se refere a outras listagens.

Outras instruções do tipo CREATE existem em SQL, mas não serão tratadas neste texto.

A instrução DROP TABLE

A instrução **DROP TABLE** simplesmente apaga uma tabela do sistema, tantos os dados como sua estrutura. Sua forma mais comum é: "DROP TABLE NOME_DA_TABELA". Um exemplo que destruiria a tabela "Local" é:

```
DROP TABLE Local;
```

5.3.2 Data Query Language

A **Data Query Language** tem como finalidade fornecer funções de consulta a base de dados.

Esse subconjunto de SQL é formado pelos comandos SHOW e SELECT, que é o comando de consulta as informação no SGDB.

O comando SELECT é um dos mais poderosos de toda a linguagem. Sua forma mais simples retorna uma tabela inteira. Por exemplo, todos as linhas da tabela "Local" seriam retornadas com a instrução:

```
SELECT * FROM Local;
```

Se apenas alguns campos forem necessários, então a consulta poderia ser:

```
SELECT site_id, terreno FROM Local;
```

Porém, o maior poder da instrução vem de algumas capacidades adicionais. A primeira é a seleção de linha por meio de valores, como em:

```
SELECT site_id, terreno FROM Local WHERE desenvoltimento = "baixo";
```

Mais ainda, é possível listar novas tabelas a partir de outras, realizando operações conhecidas como **junção**, como em:

```
SELECT Local.site_id, terreno FROM Local , Avaliacao WHERE desenvoltimento = "baixo" and
```

Para deixar clara a complexidade da instrução SELECT, o texto a seguir apresenta a sintaxe do mesmo em MySQL(Oracle Corporation, 2019b).

```
SELECT
    [ALL | DISTINCT | DISTINCTROW ]
      [HIGH_PRIORITY]
      [STRAIGHT_JOIN]
      [SQL_SMALL_RESULT] [SQL_BIG_RESULT] [SQL_BUFFER_RESULT]
      [SQL_NO_CACHE] [SQL_CALC_FOUND_ROWS]
    select_expr [, select_expr ...]
    [FROM table_references
      [PARTITION partition_list]
    [WHERE where_condition]
    [GROUP BY {col_name | expr | position}, ... [WITH ROLLUP]]
    [HAVING where_condition]
    [WINDOW window_name AS (window_spec)
        [, window_name AS (window_spec)] ...]
    [ORDER BY {col_name | expr | position}
      [ASC | DESC], ... [WITH ROLLUP]]
    [LIMIT {[offset,] row_count | row_count OFFSET offset}]
    [INTO OUTFILE 'file_name'
        [CHARACTER SET charset_name]
        export_options
      | INTO DUMPFILE 'file_name'
      | INTO var_name [, var_name]]
    [FOR {UPDATE | SHARE} [OF tbl_name [, tbl_name] ...] [NOWAIT | SKIP LOCKED]
      | LOCK IN SHARE MODE]]
```

Essa sintaxe esconde muita complexidade, principalmente nos termos `table_references` e `where_condition`.

O exemplo a seguir mostra uma sentença SELECT contendo outra cláusula SELECT detro dela, que responde com o nome de todos os especialistas que não fizeram nenhuma avaliação de mais de R$ 100.000,00 reais.

```
SELECT nome
  FROM Especialista
 WHERE nore NOT IN (SELECT nome
                      FROM Avaliacoes
                     WHERE custos > 100000);
```

5.3.3 Uma estratégia para criar consultas SQL

Tendo em vista a importância da instrução SELECT na criação de consultas SQL para o analista de dados, apresento aqui uma estratégia de 8 passos para criar um consulta:

1. escolher as tabelas (cláusula FROM);
2. juntar as tabelas em uma só (cláusulas JOIN);
3. selecionar campos (nomes dos campos);
4. agrupar (cláusula GROUP BY);
5. ordenar (cláusula ORDER BY);
6. filtrar pelos campos das tabelas (cláusula WHERE);
7. filtrar pelos campos agrupados (cláusula HAVING), e
8. limitar o tamanho da resposta (cláusula LIMIT)

Chamamos a atenção que essa ordem é arbitrária, e a construção de expressões SQL complicadas pode ser feita de maneira iterativa e incremental, de forma que fique claro para o autor que fazem o que ele deseja.

A ideia básica é saber em que tabelas estão os dados necessários, e a partir dessas tabelas construir a consulta necessária.

Por exemplo, vamos supor que está sendo usada o banco de dados da base SAKILA, apresentada no capítulo 4, na figura 4.2.

Vamos supor que a consulta desejada pode ser descrita como: listar o valor total obtido com aluguel para cada filme com rating "R", caso o valor seja maior que 100 dólares, ordenado por valor total, para os 10 filmes que mais arrecadaram.

Escolher as tabelas

Uma observação do modelo da base indica que será necessária usar as tabelas: film, inventory, rental e payment. Nosso esqueleto de consulta fica sendo: SELECT * FROM FILM, INVENTORY, RENTAL , PAYMENT. Essa consulta não executa ainda.

Juntar as tabelas em uma só

O segundo passo é juntar as tabelas. Isso pode ser feito com a clásula WHERE, mas prefirimos usar as junções (JOINs). Quatro tipo de junções são possíveis: INNER, LEFT, RIGHT e FULL[3].

O INNER JOIN exige que cada linha das tabelas juntadas tenham valores que casam, sendo o *join* mais comum, e não aceitando os nulos. O outros garantem que todos as linhas da tabela da esquerda, da direita e das duas apareçam, mesmo que não haja um casamento perfeito.

No nosso caso queremos juntar as tabelas em uma ordem, criando uma consulta já executável:

```
SELECT *  FROM film
INNER JOIN inventory ON film.film_id=inventory.film_id
```

[3]Elas também são conhecidas como LEFT OUTER, RIGHT OUTER e FULL OUTER.

```
INNER JOIN rental ON inventory.inventory_id=rental.inventory_id
INNER JOIN payment ON rental.rental_id=payment.rental_id;
```

Selecionar campos

Vamos ficar apenas com o título do filme e o valor recebido como pagamento, criando a seguinte consulta:

```
SELECT film.title , amount  FROM film
INNER JOIN inventory ON film.film_id=inventory.film_id
INNER JOIN rental ON inventory.inventory_id=rental.inventory_id
INNER JOIN payment ON rental.rental_id=payment.rental_id;
```

A seleção de campos permite também a mudança de nomes, como por exemplo em:

```
SELECT film.title as titulo FROM film
```

Agrupar se necessário

Nós queremos agrupar, por valor total de todas os aluguéis, então nossa consulta passa a ser:

```
SELECT film.title , SUM(amount) as total  FROM film
INNER JOIN inventory ON film.film_id=inventory.film_id
INNER JOIN rental ON inventory.inventory_id=rental.inventory_id
INNER JOIN payment ON rental.rental_id=payment.rental_id
GROUP BY film.title;
```

A cláusula GROUP BY pode ser feita com várias colunas. Normalmente as colunas não agrupadas que aparecem na seleção devem ser alteradas para conter uma operação de agregação, como SUM, MAX,etc.

Ordenar se necessário

Vamos ordenar pelo total, em ordem descendente:

```
SELECT film.title , SUM(amount) as total  FROM film
INNER JOIN inventory ON film.film_id=inventory.film_id
INNER JOIN rental ON inventory.inventory_id=rental.inventory_id
INNER JOIN payment ON rental.rental_id=payment.rental_id
GROUP BY film.title
ORDER BY total DESC;
```

A cláusula ORDER BY também permite o uso de várias colunas, e a ordem será usada na ordenação, e pode ser feita de forma descendente (DESC) ou ascendente (ASC)

Filtrar se necessário

Vamos tratar só de filmes com rating igual a "R":

```
SELECT film.title , rating , SUM(amount) as total  FROM film
INNER JOIN inventory ON film.film_id=inventory.film_id
INNER JOIN rental ON inventory.inventory_id=rental.inventory_id
INNER JOIN payment ON rental.rental_id=payment.rental_id
WHERE film.rating = "R"
GROUP BY film.title
ORDER BY total DESC;
```

A cláusula WHERE é das mais poderosas do SELECT, permitindo uma longa construção de operações lógicas, sendo também usada para fazer INNER JOINs implicitamente. Devemos tomar cuidado porque campos criados com operações de agregação são filtrados com a clásula HAVING.

Filtrar os valores agrupados se necessário

Vamos filtrar para os totais maiores que 100 dólares:

```
SELECT film.title , SUM(amount) as total  FROM film
INNER JOIN inventory ON film.film_id=inventory.film_id
INNER JOIN rental ON inventory.inventory_id=rental.inventory_id
INNER JOIN payment ON rental.rental_id=payment.rental_id
WHERE film.rating = "R"
GROUP BY film.title
HAVING total>100
ORDER BY total DESC;
```

A cláusula HAVING é dedicada a filtrar por campos criados com funções de agregação.

Limitar o tamanho da saída

Limitando a 10 resultados:

```
SELECT film.title , SUM(amount) as total  FROM film
INNER JOIN inventory ON film.film_id=inventory.film_id
INNER JOIN rental ON inventory.inventory_id=rental.inventory_id
INNER JOIN payment ON rental.rental_id=payment.rental_id
WHERE film.rating = "R"
GROUP BY film.title HAVING total>100
ORDER BY total DESC
LIMIT 10;
```

A cláusula LIMIT é auto explicativa.

5.3.4 Data Manipulation Language

Esse subconjunto de SQL é formado pelos comandos INSERT, UPDATE, DELETE, MERGE, CALL, EXPLAIN PLAN e LOCK TABLE.

Neste texto serão tratados os comandos INSERT, UPDATE e DELETE são responsáveis por colocar, alterar e apagar informações das tabelas de um SGDB.

Essas instruções alteram as tabelas e nunca devem ser usadas por um analista de dados nas tabelas fontes. O ideal é que ele não tenha nem mesmo permissão para fazê-lo, a não ser em tarefas de correção de dados e após todos os testes necessários e um backup de segurança ter sido feito antes da operação.

DELETE

A instrução delete é a mais simples de definir. Para isso, basta normalmente usar algo como:

```
DELETE FROM Local WHERE terreno = "Arenoso";
```

Sua sintaxe completa no MySQL é(Oracle Corporation, 2019b):

```
DELETE [LOW_PRIORITY] [QUICK] [IGNORE] FROM tbl_name [[AS] tbl_alias]
    [PARTITION (partition_name [, partition_name] ...)]
    [WHERE where_condition]
    [ORDER BY ...]
    [LIMIT row_count]
```

Cuidado, porém, pois um DELETE errado pode fazer você perder os dados totalmente, a menos da existência de um backup. Não recomendo apagar nenhum dado nas fontes, na verdade, recomendo que um analista de dados nem tenha a permissão para fazer isso no SGDB, para evitar erros.

INSERT

O objetivo dessa instrução é inserir dados em uma tabela. Sua forma mais usada é simplesmente inserir uma linha em uma tabela listando os valores de todos os seus campos na ordem correta(Oracle Corporation, 2019b), como em:

```
INSERT INTO Local VALUES("s123-a","pantanoso","alto");
```

Sua sintaxe, em MySQL(Oracle Corporation, 2019b), permite variações:

```
INSERT [LOW_PRIORITY | DELAYED | HIGH_PRIORITY] [IGNORE]
    [INTO] tbl_name
    [PARTITION (partition_name [, partition_name] ...)]
    [(col_name [, col_name] ...)]
```

```
    {VALUES | VALUE} (value_list) [, (value_list)] ...
    [ON DUPLICATE KEY UPDATE assignment_list]

INSERT [LOW_PRIORITY | DELAYED | HIGH_PRIORITY] [IGNORE]
    [INTO] tbl_name
    [PARTITION (partition_name [, partition_name] ...)]
    SET assignment_list
    [ON DUPLICATE KEY UPDATE assignment_list]

INSERT [LOW_PRIORITY | HIGH_PRIORITY] [IGNORE]
    [INTO] tbl_name
    [PARTITION (partition_name [, partition_name] ...)]
    [(col_name [, col_name] ...)]
    SELECT ...
    [ON DUPLICATE KEY UPDATE assignment_list]

value:
    {expr | DEFAULT}

value_list:
    value [, value] ...

assignment:
    col_name = value

assignment_list:
    assignment [, assignment] ...
```

UPDATE

A instrução update permite alterar o valor de células específicas de acordo com uma condição. Por exemplo para alterar o valor do custo de descontaminação do site L171 em mais 10%, o seguinte comando seria válido.

```
UPDATE Avaliacoes SET custos=custos*1.1 WHERE site_id="L171";
```

A sintaxe completa da instrução UPDATE para MySQL(Oracle Corporation, 2019b) é:

```
UPDATE [LOW_PRIORITY] [IGNORE] table_reference
    SET assignment_list
    [WHERE where_condition]
    [ORDER BY ...]
    [LIMIT row_count]
```

```
value:
    {expr | DEFAULT}

assignment:
    col_name = value

assignment_list:
    assignment [, assignment]

UPDATE [LOW_PRIORITY] [IGNORE] table_references
    SET assignment_list
    [WHERE where_condition]
```

5.3.5 Data Transaction Language

Esse subconjunto de SQL é formado pelos comandos BEGIN TRANSACTION, COMMIT E ROLLBACK. Esses comandos permitem que o início, o fim e ainda abortar uma transação composta de várias operações no banco de dados.

5.3.6 Data Control Language

Esse subconjunto de SQL é formado pelos comandos GRANT, DENY e REVOKE. Esses comandos controlam os privilégios de acesso às tabelas do SGDB e não são tratados neste texto.

5.4 O que mais

Existe muito mais a falar sobre bancos de dados relacionais e SQL. Milhares de textos já foram escritos sobre o assunto. Este capítulo tem como finalidade apenas ser uma visão rápida, em especial para o leitor que já viu o tema e pode ter se esquecido de algumas coisas.

Havendo mais interesse, os livros *Fundamentals of Database Systems* de Elmasri e Navathe (2016), *An introduction to database systems* de Date (2004) e *Joe Celko's SQL for smarties: advanced SQL programming* Celko (2005) são boas referências. Uma boa introdução a SQL pode ser encontrada também no site W3School (`https://www.w3schools.com/sql`). Há muito mais material de boa qualidade disponível em livros, artigos, tutoriais e sites.

6 A Situação das Informações nas Organizações

6.1 A necessidade de informação das organizações

Todas as organizações têm a necessidade de controlar e gerenciar todas as informações que dispõem sobre si mesmas e sobre o mercado, de modo a permitir que suas decisões produzam ações que gerem benefícios e evitem prejuízos, de curto, médio e longo prazo, nos níveis operacional, tático e estratégico.

As informações necessárias para a organização aparecem em todo seu ambiente. Cada ação realizada, cada decisão tomada, gera, ou pelo menos deveria gerar, um registro que pode ser utilizado mais tarde para a tomada de decisões. Por exemplo, cada venda feita por uma cadeia de lojas de varejo pode ser incluída em um gráfico ou relatório que permita entender o desempenho das vendas por local, por produto, por vendedor e até mesmo pela hora do dia. Em um uso mais avançado, esses dados podem ser correlacionados, com o processo de seleção da área de Recursos Humanos, de maneira a tentar desenvolver um perfil do vendedor a ser contratado, tanto de forma geral como para atender necessidades específicas. Com relação aos dados de mercado, podem ser usados para prever o desempenho de uma nova loja em uma localidade específica, de acordo com dados demográficos ou informações sobre outros tipos de comércio ou mesmo de consumo de água ou eletricidade.

Para isso ser possível é necessário informatizar a organização, isto é, implantar dentro da organização sistemas de informação que gerenciem essas informações.

6.2 Processo de Informatização nas Organizações

Frente a importância da informação para o sucesso da organização, é fácil entender que, ao longo de sua vida, ela deve passar por um processo contínuo de informatização, a fim de atender as demandas cada vez maiores tanto da própria organização, quanto de seus fornecedores, parceiros, clientes e também do Estado.

Esse processo é necessário e geralmente benéfico, mas não sem problemas.

Entre os principais problemas do processo de informatização das organizações é que ele normalmente é de crescimento vegetativo, *bottom-up* e por demanda imediata. Os sistemas são criados um a um, em função de necessidades específicas trazidas pelas partes interessadas, e construídos de modo a atender principalmente os requisitos dessas partes, muitas vezes desconsiderando como as atividades e informações desses sistemas podem ser úteis para o resto da organização.

Por outro lado, tentativas de criar grandes sistemas integrados que atendam toda a organização parecem estar fadados ao fracasso. O tamanho de um sistema é um fortíssimo fator de risco(Pressman e Maxim, 2016), já que a relação do tamanho com a taxa de fracasso é mais que linear.

Como sempre, as melhores soluções, ou pelos menos as soluções mais viáveis, parecem estar em um meio termo, ou pelo menos em decisões tomadas de forma consciente dos riscos e benefícios de cada opção.

Não se pode também deixar de levar em consideração que organizações diferentes têm gestões diferentes desse processo. Enquanto uma pode escolher os sistemas de informação a serem implantados de forma *ad-hoc*, outra, mais corretamente, pode possuir um planejamento estratégico que indica em que direção a área de Tecnologia da Informação deve seguir e que sistemas serão estratégicos para o seu progresso.

Esse quadro bastante variado, evoluiu muito tecnologicamente e no entendimento do negócio informatizado nos últimos anos. Alguns grandes marcos, como o aparecimento do computador comercial, o aparecimento do computador pessoal, o aparecimento das redes locais e finalmente o aparecimento da Internet e das tecnologias em nuvem, causaram grandes mudanças na forma de pensar a organização, criando até mesmo a possibilidade de organizações virtuais.

Ao logo desse tempo houve um aprendizado que trouxe soluções bastante adequadas ao problema de gestão das organizações, como o uso de sistemas **ERP** e a compra de software **COTS**.

COTS significa **Commercial Of The Shelf**, e indica software pronto que fornece funções específicas com nenhuma ou pouca adaptação para quem o compra ou implementa. Principalmente pequenos negócios e profissionais liberais podem ter quase todas suas necessidades atendidas por meio de esse tipo de software. Algumas áreas de aplicação, mesmo em empresas grandes, podem se beneficiar desse tipo de produto. Deve ficar

claro, porém, que nesse caso a organização deve se adaptar as práticas implementadas no produto.

Já os sistemas **ERP**, **Entreprise Resource Planning**, são na prática sistemas de gestão empresarial customizáveis destinados a integrar fortemente as áreas de negócio. Sua principal característica é funcionar em áreas do negócio cuja tarefa é bem definida, e cuja experiência de desenvolvimento de software é muito forte, e seu maior sucesso vem da implementação, já no software, das melhores práticas do negócio.

Soluções ERP hoje são capazes de servir grande parte das necessidades genéricas de todas as empresas, desde funções de Recursos Humanos até detalhes de Contabilidade, sendo que algumas empresas inclusive mudam seus processos para se adaptar mais facilmente a um ERP específico.

Essas soluções, porém, são genéricas. Mesmo quando customizáveis, dificilmente atendem necessidades importantes dos processos das cadeias de valor das empresas, pois estes são específicos e poucas empresas desejam remodelá-los de acordo com uma prática comum no mercado, até mesmo porque suas diferenças podem fornecer justamente o diferencial competitivo que possuem. Assim, o software desenvolvido sob encomenda, interna ou externamente, ainda possui, e certamente sempre possuirá, um presença marcante em todos as organizações.

Todos esses sistemas geram dados. Quase que escondidos nesses dados estão as informações necessárias para a tomada de decisão.

6.3 A necessidade centralização dos dados

O outro processo de grande sucesso foi a centralização dos dados dos sistemas operacionais e transacionais (OLTP) em bases de dados integradas na empresa, principalmente em SGDBs Relacionais de grande porte.

O quadro do primeiro quarto do século XXI é que a maioria das organizações entende que seus dados operacionais devem estar sob controle centralizado. O ideal é que todos esses sistemas compartilhem uma mesma base, porém necessidades como desempenho, custo de implementação, estratégia empresarial, permanência da tecnologia usada no mercado, e ainda outras, podem levar a existência de mais de uma base.

Esta situação também não é sem seus problemas. A centralização dos dados sob um controle único pode tanto auxiliar como trazer dificuldades para o desenvolvimento rápido de sistemas, principalmente quando há exigência de muita burocracia para alterar a base ou estruturar informações de forma a atender novas demandas de negócio.

O uso de softwares COTS ou ERP integrados, chave do sucesso para parte da operação da empresa, pode criar uma forte dependência da organização com um vendedor, de forma que seja difícil fazer decisões estratégicas de mudança de produto sem um grande

custo para a organização, ou simplesmente que seja difícil extrair os dados desses sistemas para atender outras demandas.

Em todo caso, antes da proposta dos Data Warehouse a situação das informações dentro das organizações podia ser descrita, de forma geral, da seguinte maneira:

- Reconhecimento da necessidade de centralização dos dados operacionais, nas aplicações do tipo OLTP, normalmente na forma do uso de uma base de dados unificada, ou de várias bases controlados por um serviço central.
- Existência de vários sistemas de informação, centralizados ou não. Os sistemas não centralizados existem tanto por motivos históricos, no caso de software legado, quanto de aplicações que foram desenvolvidas fora do sistema padrão por diversos motivos: desenvolvimento distribuído nas partes da organização, necessidade de passar por cima de processos burocráticos para atender demandas imediatas de negócio, necessidade de uso de tecnologias específicas, etc.
- Existência de uma grande quantidade de sistemas criados pelos próprios usuários acima do nível operacional, muitas vezes na forma de planilhas eletrônicas, documentos ou mesmo bancos de dados simples, como o Microsoft Access.
- Pouca integração dos dados e dos sistemas destinado ao apoio da decisão, geralmente criados de forma *ad-hoc* e possivelmente a partir de retratos diferentes da informação disponível na organização.

Esse últimos dois fatores levaram a um quadro caótico nos dados destinados a tomada de decisão dentro da organização. Enquanto os dados operacionais eram mantidos dentro de algum controle, especialmente porque são essenciais para o funcionamento da organização, os dados usados para a tomada de decisão, geralmente sumários ou recortes dos dados transacionais, eram baseados em fotografias tiradas de forma diferente e em momentos diferentes da empresa. Esse é o principal diagnóstico que leva a necessidade da existência de Data Warehouses por Inmon (2005): a falta de organização e o excesso de redundância e incertezas nesses dados.

Um exemplo do que poderia acontecer dentro da organização por causa desta situação caótica são dois gerentes ou diretores chegarem a uma reunião com números diferentes sobre um mesmo objetivo da empresa, devido a formas como esses dados foram obtidos. Um diretor, por exemplo, com os dados de venda por semestre, até o último semestre completo, poderia dizer que as vendas estavam aumentando, enquanto um segundo diretor, com os dados de vendas por mês poderia dizer que estavam diminuindo. Dependendo da fonte da informação, a confusão podia ser maior. Um diretor, por exemplo, poderia estar contabilizando os pedidos, enquanto outro os pagamentos feitos. Ainda, um diretor poderia ter usado dados obtidos com algum viés de seleção, como desconsiderar um canal, ou considerar apenas as vendas para os grandes clientes.

A verdade é que se os dados são muitas vezes extraídos dos sistemas OLTP de forma arbitrária e depois trabalhados manualmente pelos membros da organização. A tendência é que se espalhem, sejam alterados em formato e conteúdo e criem uma miríade de fotografias que não só mostram realidades diferentes da empresa, mas que também foram

tiradas de forma diferente, com motivações diferentes e que não podem ser mapeadas ao processo original de extração.

Para dar um ideia do problema do espalhamento de dados em um organização, que ocorre tipicamente por planilhas eletrônicas, uma revisão de diferentes pesquisas sobre erros em planilhas de 1995 a 2008 calculou erros em 88% das planilhas no total, sendo que algumas dessas pesquisas encontrou erros em 100% das planilhas investigadasPanko (1998). Soto (2019) relata erros específicos em planilhas que causaram prejuízos a organizações, como a venda de 10.000 ingressos não existentes para os Jogos Olímpicos de Londres de 2012, obrigando o Comitê a substitui-los por outros ingressos de maior valor.

Todo esse quadro anteriormente exposto mostra que apesar dos enormes esforços e algo custo da gestão das informações nas empresas, mesmo em 2019, na prática essa informação ainda é muito carente de organização e controle.

Mesmo depois de todo o aprendizado que foi obtido, de todas as propostas feitas e adotadas, a verdade é que nas organizações do mundo real acaba levando a soluções não ideais. O resultado é que organização acaba por tomar decisões importantes sobre dados com pouca qualidade.

6.4 Momento Histórico da Proposta dos Data Warehouse

Em outra escala, o problema do descontrole dos dados já ocorria na década de 1970, onde começaram a aparecer os primeiros sistemas de informação automatizados.

A partir de experiência em trabalhos semelhantes ao que hoje é chamado de Data Warehouse, em 1992 Inmon (1992) lançou seu primeiro livro sobre o assunto, chamando a atenção de todos para a necessidade de aumentar a qualidade dessa dado, na forma do que ele denominou de Data Warehouse. Nessa época, os microcomputadores, as planilhas e os sistemas criados diretamente pelo usuário já eram parte integradas da Tecnologia da Informação nas organizações. Porém, ainda não tinha sido atingida a fase do **Big Data**, e tantas outras tecnologias comuns hoje em dia não estavam disponíveis.

Mesmo assim, o retrato apresentado por Inmon (1992) e renovado em Inmon (2005) pouco difere do retrato que vemos agora. Se algo aconteceu foi o aparecimento de mais fontes de dados e de demandas cada vez maiores de tratar informação.

6.5 O Data Warehouse

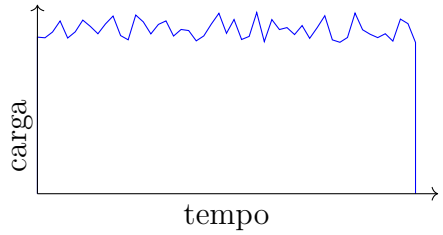
Então, da mesma forma que as bases relacionais propiciam a integração dos dados transacionais, Inmon (2005) e outros perceberam que era necessário **centralizar** os dados destinados a tomada de decisão, normalmente por meio de sistemas OLAP. Mais que isso, que esses dados não deveriam estar registrados da mesma forma dos sistemas OLTP, pois

eles refletem uma outra visão, que é composta de dados históricos, persistentes organizados pelo assunto, e com outras características que vão definir um **Data Warehouse**.

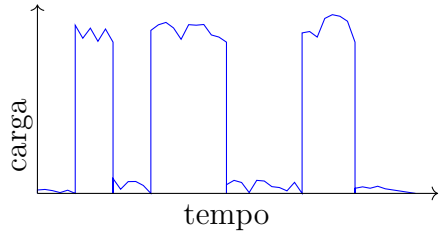
De certa forma, o Data Warehouse, como conceito, está para os sistemas OLAP e de tomada de decisão assim como as Bases de Dados Integradas estão para os sistemas OLTP e operacionais. Trazer toda a informação para um ambiente controlado, sanitizado e centralizado é uma virtuda a ser perseguida.

Para ser implementado e ser útil, porém, o Data Warehouse precisa ser adaptado ao fato que a base integrada organizacional é apenas uma utopia. Por isso, a implantação de um Data Warehouse também precisa estabelecer uma metodologia de coleta e transformação dos dados. Para isso ele tem que ter acesso as várias fontes de informação primitiva da empresa e a capacidade de transformá-la e armazená-la de forma a atender, de maneira eficiente, as demandas dos sistemas OLAP e de tomada de decisão. Isso vai levar a existência de uma estrutura complexa que inclui mecanismos que usam nomes como *ETL*, *ELT* e *Staging*, tratados neste livro.

O Data Warehouse também tem que ser separado dos sistemas OLTP por questões de desempenho. A forma de acesso as operações diárias gera uma carga no sistema, e exigências de tempo de atendimento, diferente das operações realizadas em um Data Warehouse(Inmon, 2005). A Figura 6.1 mostra a diferença do comportamente da carga desses dois sistemas.



(a) OLTP



(b) OLAP

Figura 6.1: Diferentes cargas exigidas de sistemas OLTP e OLAP. Fonte: (Inmon, 2005)

Finalmente, para corretamente distribuir os dados a quem precisa e pode consultá-los, levando em consideração fatores tanto de desempenho quanto como leis como a Sarbanes–Oxley(United States Code, 2002), que fala do controle do acesso à informação

dentro da organização, o Data Warehouse precisa alimentar bases menores conhecidas como **Data Marts**.

6.6 Data Warehouse e Bancos de Dados

O mercado atual usa o termo Banco de Dados, majoritariamente querendo dizer que é relacional, para indicar o repositório de dados da organização, em especial o repositório de dados OLTP, visando o rápido acesso a dados únicos, para acelerar essas transações. É possível e desejado fazer relatórios e gráficos a partir dos dados nos bancos de dados, porém eles normalmente refletem o estado atual da organização.

O termo Data Warehouse é reservado para indicar o repositórios de dados históricos da organização, visando o rápido acesso a visões agregadas, como relatórios, e servir operação OLAP.

Outra diferença básica é que o modelo de dados de um Banco de Dados é criado para permitir o uso mais geral possível dos dados, e segue regras, conhecidas como normalização, para evitar redundâncias. Já o Data Warehouse é criado para atender necessidades específicas, com um modelo muitas vezes não normalizado.

Porém, há um erro em tentar criar essa dicotomia. A verdade é que um Data warehouse precisa de um sistema de banco de dados para funcionar, ou seja, os Data Warehouse tem que existir dentro de um sistema de informação com as mesmas características dos sistemas de banco de dados. Devido a existência de SGBD relacionais de alto desempenho, que com o tempo foram adaptados para atender melhor as demandas de desempenho de um Data Warehouse, grande parte dos Data Warehouse funciona dentro de sistemas relacionais.

Existem porém outras bases não relacionais muito utilizadas para Data Warehouse. Normalmente elas se caracterizam por serem otimizadas para certo tipo de consultas, normalmente operações sumários em uma tabela e operações OLAP, e no uso do Modelo Dimensional, que será tratado mais tarde neste texto.

6.7 Data Warehouses ainda são necessárias?

Tendo em vista que Data Warehouses foram projetadas no final do século XX, é possível questionar sua aplicabilidade quase 30 anos depois de seu aparecimento.

Uma questão que se põe é causada pelo avanço da tecnologia. A disponibilidade de armazenar um volume muito maior em discos e o uso de múltiplas CPUs nos servidores permite que sejam construídos sistemas de informação OLTP que pouco se preocupam com a necessidade de limpar periodicamente os dados antigos do sistema. Esses dados podem então ser acessados diretamente?

A resposta para essa questão está no fato que sistemas OLTP e OLAP, e também as novas aplicações de **Business Intelligence** (**BI**), trazem requisitos diferentes, que podem ser atendidos exatamente pelos Data Warehouses.

Por outro lado, sistemas de *Big Data* e *Data Lake* podem parecer concorrentes ao sistemas de Data Warehouse, quando de fato são complementares e acabam criando uma sinergia.

6.8 Mais de um Data Warehouse?

Apesar da ideia básica do Data Warehouse é ser uma base central de informações, como nos casos dos sistemas de informação tradicionais, fatores do dia a dia fizeram que o sonho do Data Warehouse centralizado fosse substituído, no mercado, pela existência, dentro da organização, de mais de um Data Warehouse. Uma pesquisa descrita por Wells e Nahari (2019) mostrou que a maioria das empresas usa mais de um Data Warehouse, de acordo com os dados apresentados na Figura 6.2.

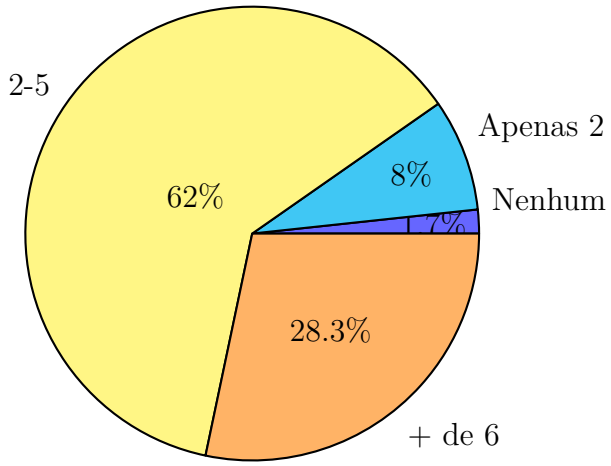


Figura 6.2: Número de data warehouses por empresa(Wells e Nahari, 2019).

7 Data Warehouse

7.1 O Debate Kimball x Inmon

Dois autores dominam as publicações sobre Data Warehouse no mundo.

O primeiro deles é **Bill Inmon**, considerado o pai do Data Warehouse, que começou a discutir o assunto e cunhou o termo ainda em 1970. Seu livro *Building the Data Warehouse*, em sua primeira edição de 1992, foi um marco importante na divulgação do tema(Kempe e Williams, 2012).

O segundo autor é **Ralph Kimball**. que com a primeira edição de *The Data Warehouse Toolkit*, de 1996, trouxe um conjunto de práticas e exemplos que facilitaram a implementação de data warehouses para muitos(Kempe e Williams, 2012).

Inmon e Kimball mostram caminhos diferentes de implementar data warehouses, sendo que Inmon favorece a abordagem *top-down*, baseada em um modelo normalizado, e Kimball a abordagem *bottom-up*, baseada em um modelo dimensional, de criação de data warehouses. A polêmica entre suas ideias é uma das mais conhecidas da Informática.

Em especial, Ralph Kimball (2013) apresentou ao público e defende o uso do do Modelo Dimensional[1]. Ele diz: "*Data in the queryable presentation area of the data warehouse must be dimensional*"(Ralph Kimball, 2013). Já Inmon (2005) diz "*Star schemas are not very good in the long run for a data warehouse.*" Esquemas Estrela são a forma de implementar modelos dimensionais em Bancos de Dados Relacionais.

[1] O Modelo Dimensional e os termos Fato e Dimensão foram inventados nos anos 1960 por um projeto conjunto da *Dartmouth University* e da *General Mills*(Ralph Kimball, 2013)

7.2 Definição de Inmon para Data Warehouse

Segundo Inmon (2005), um **Data Warehouse** é "Uma coleção de dados orientada a assunto, integrada, não volátil e variante no tempo que apoia as decisões de gerência."

7.2.1 Orientação a Assunto

A ideia de Inmon é que enquanto os sistemas tradicionais são organizados ao redor das funções da empresa, ou seja, do que a empresa faz, os data warehouses são organizados em torno dos **assuntos** que a empresa trata, ou seja, dos tópicos sobre as quais ela tem que tomar decisão.

Por exemplo, o que faz um mercado de varejo? Em sua cadeia de valor, Figura 7.1, ele compra no atacado, estoca e vende produtos no varejo. Os seus sistemas de informação são orientados para essas atividades. Uma lista de sistemas de informação que podem estar disponíveis em um pequeno mercado é mostrada na Tabela 7.1.

Tabela 7.1: Sistemas em um pequeno mercado.

Sistemas do Mercado
Controle de Estoque
Contas a Pagar/Receber
Controle do Fiado
Caixa Registradora
Pedidos por Telefone
Entregas

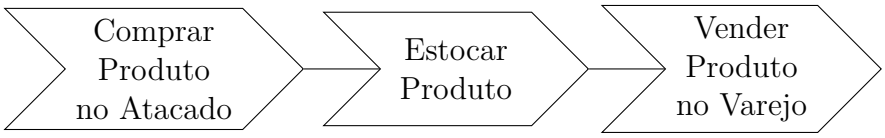


Figura 7.1: A cadeia de valor de um mercado, descrição em ARIS.

Porém, como o mercado analisa suas informações por assuntos? Por exemplo, com perguntas como "vendas por produto" ou "vendas por loja". O assunto, nesse caso, é **vendas**, e a informação principal é um agregado de várias instâncias do banco de dados que registram um tipo de venda.

Nas fontes originais de dados dentro da empresa, a informação referente a um assunto pode estar dividida de várias formas. Por exemplo, para o assunto cliente podemos ter seu nome e CPF em um registro de uma venda, porém seu endereço só estará em um registro de uma entrega. Já seu telefone pode estar em um registro de um pedido feito pelo telefone para entrega. Todas essas ações são atendidas por um ou mais sistemas de informação da empresa, e os dados podem estar em um ou mais repositórios, e mesmo

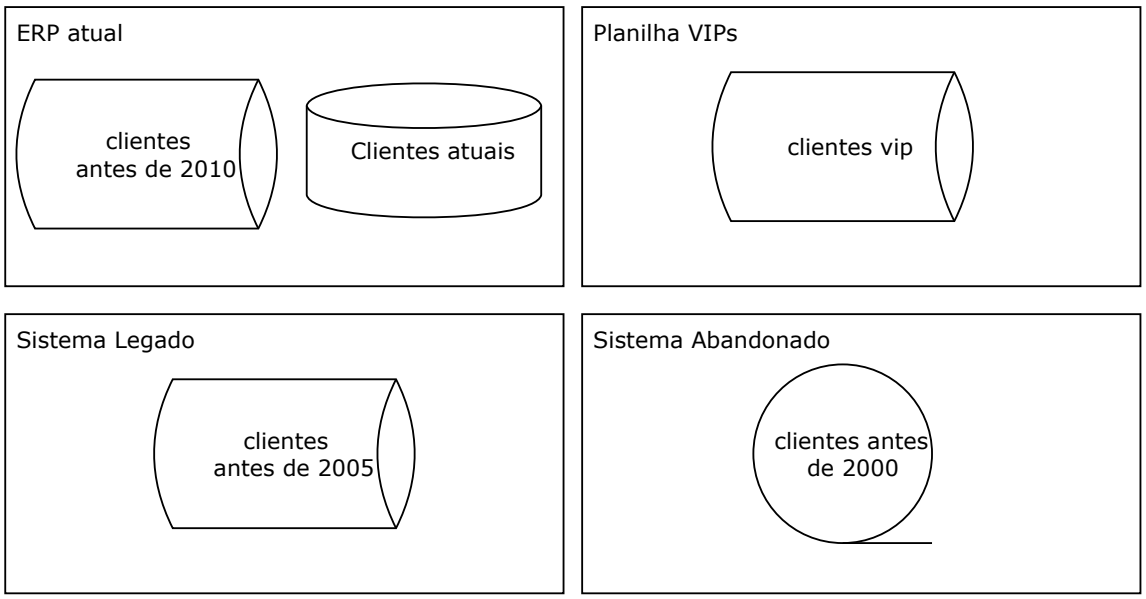


Figura 7.2: Diversas fontes de onde pode ser obtido o assunto cliente

dentro de tabelas diferentes em uma base de dados única, dependendo da maturidade da empresa no tratamento de seus dados.

Além disso, tanto os sistemas são trocados e evoluem com o tempo, quanto os dados e seus formatos.

Por exemplo, para o assunto cliente, podemos ter dados dos clientes atuais em um SGDB, dados de clientes antigos guardados em fitas backup de sistemas legados, dados de clientes de um setor específico guardados em uma planilha, etc.

Podemos também ter dados repetidos de um mesmo cliente, e com algumas alterações. Um cliente que mudou de endereço, ou mesmo uma cliente que trocou de nome ao casar(Inmon, 2005). A Figura 7.2 mostra um exemplo de fontes possíveis sobre clientes.

Cada assunto define um conjunto de tabelas correlacionadas (Inmon, 2005). No modelo dimensional (Ralph Kimball, 2013), cada assunto vai definir um (ou mais) modelos compostos de uma tabela central, que indica o tópico consultado, a **tabela fato**, e tabelas auxiliares que mostram como a tabela fato pode ser consultada, as **tabelas dimensão**.

É importante lembrar que quando trazidas para o data warehouse, todas essas informações sobre o cliente tem que ser processadas de forma que um mesmo cliente seja sempre identificado por uma mesma chave.

8 OLAP

OLAP é a sigla para **Online Analytical Processing**. O termo é usado para diferenciar o tipo de sistema, ou de operações, que se usa quando o objetivo é analisar os dados de uma organização, normalmente para a tomada de decisão, do uso normal dos dados dos sistemas que suportam o funcionamento da organização, conhecidos como transacionais ou **Online Transation Processing**, **OLTP**, ou ainda **processamento de transações em tempo real**.

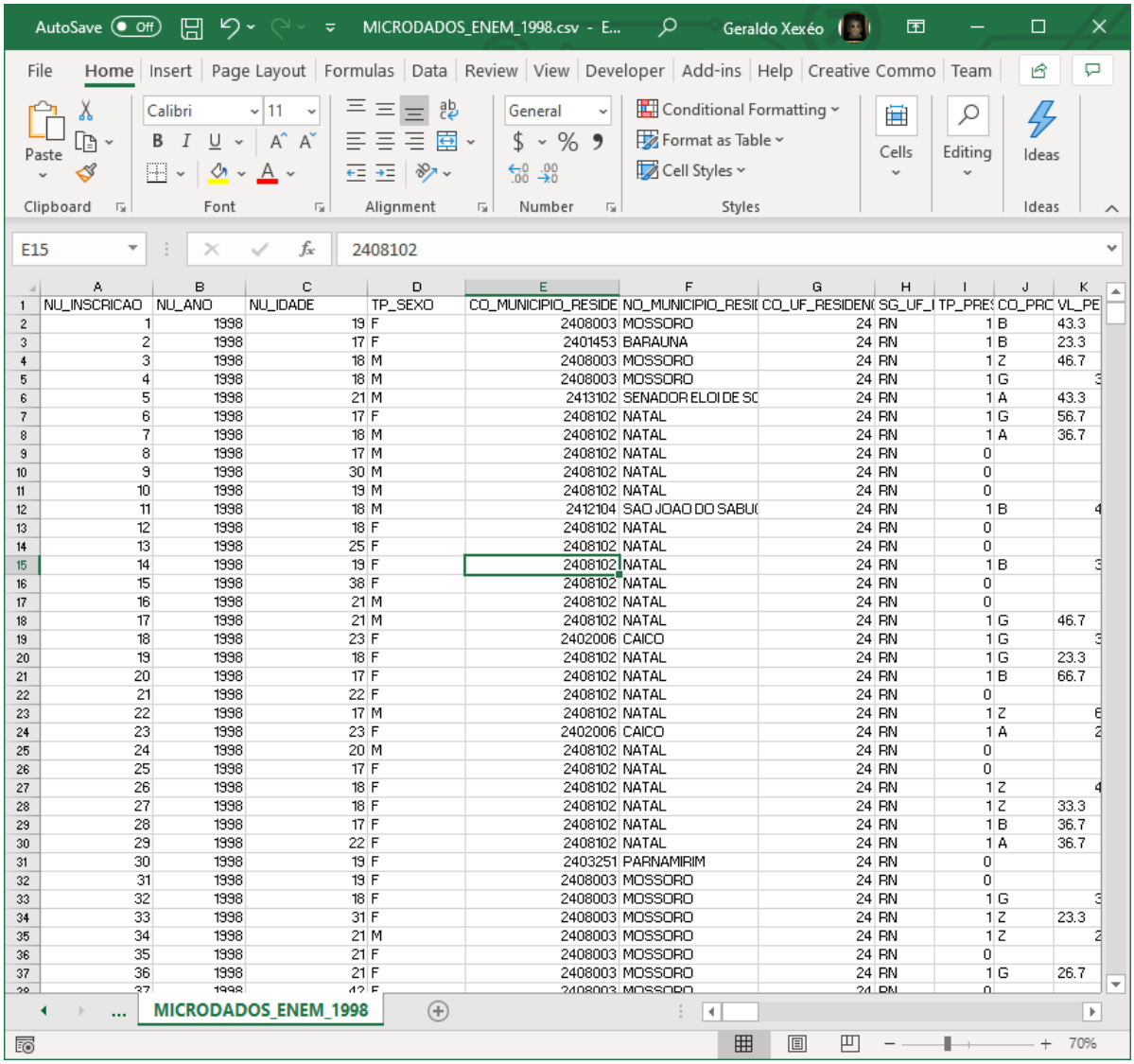
Assim, sistemas OLTP e sistemas OLAP, dentro de uma organização, são normalmente usados em momentos diferentes, por usuários diferentes e com objetivos diferentes. Além disso, guardam tipos de dados diferentes.

Sistemas OLAP são tipicamente construídos sobre data warehouses, mas não necessariamente. Além disso, eles admitem uma série de operações, conhecidas como **operações OLAP**, que permitem navegar e analisar os dados, normalmente por meio de um programa de computador com interface de usuário bastante dinâmica.

As principais operações OLAP são o **slice and dice** e o **drill down and roll up**. Além disso são também conhecidas as operações **pivot** e **drill across**.

Para ilustrar algumas operações, neste capítulo serão usados como dados exemplos uma planilha com os micro-dados do ENEM 1998[1](Estudos e Pesquisas Educacionais Anísio Teixeira INEP, 2016) dentro do software Microsoft Excel™(Microsoft, 2019).

[1]Esta planilha tem poucos candidatos e poucas colunas



	A	B	C	D	E	F	G	H	I	J	K
1	NU_INSCRICAO	NU_ANO	NU_IDADE	TP_SEXO	CO_MUNICIPIO_RESIDE	NO_MUNICIPIO_RESI	CO_UF_RESIDEN	SG_UF_I	TP_PRE	CO_PRC	VL_PE
2	1	1998	19	F	2408003	MOSSORO	24	RN	1	B	43.3
3	2	1998	17	F	2401453	BARAUNA	24	RN	1	B	23.3
4	3	1998	18	M	2408003	MOSSORO	24	RN	1	Z	46.7
5	4	1998	18	M	2408003	MOSSORO	24	RN	1	G	
6	5	1998	21	M	2413102	SENADOR ELOI DE SC	24	RN	1	A	43.3
7	6	1998	17	F	2408102	NATAL	24	RN	1	G	56.7
8	7	1998	18	M	2408102	NATAL	24	RN	1	A	36.7
9	8	1998	17	M	2408102	NATAL	24	RN	0		
10	9	1998	30	M	2408102	NATAL	24	RN	0		
11	10	1998	19	M	2408102	NATAL	24	RN	0		
12	11	1998	18	M	2412104	SAO JOAO DO SABU	24	RN	1	B	
13	12	1998	18	F	2408102	NATAL	24	RN	0		
14	13	1998	25	F	2408102	NATAL	24	RN	0		
15	14	1998	19	F	2408102	NATAL	24	RN	1	B	
16	15	1998	38	F	2408102	NATAL	24	RN	0		
17	16	1998	21	M	2408102	NATAL	24	RN	0		
18	17	1998	21	M	2408102	NATAL	24	RN	1	G	46.7
19	18	1998	23	F	2402006	CAICO	24	RN	1	G	
20	19	1998	18	F	2408102	NATAL	24	RN	1	G	23.3
21	20	1998	17	F	2408102	NATAL	24	RN	1	B	66.7
22	21	1998	22	F	2408102	NATAL	24	RN	0		
23	22	1998	17	M	2408102	NATAL	24	RN	1	Z	
24	23	1998	23	F	2402006	CAICO	24	RN	1	A	
25	24	1998	20	M	2408102	NATAL	24	RN	0		
26	25	1998	17	F	2408102	NATAL	24	RN	0		
27	26	1998	18	F	2408102	NATAL	24	RN	1	Z	
28	27	1998	18	F	2408102	NATAL	24	RN	1	Z	33.3
29	28	1998	17	F	2408102	NATAL	24	RN	1	B	36.7
30	29	1998	22	F	2408102	NATAL	24	RN	1	A	36.7
31	30	1998	19	F	2403251	PARNAMIRIM	24	RN	0		
32	31	1998	19	F	2408003	MOSSORO	24	RN	0		
33	32	1998	18	F	2408003	MOSSORO	24	RN	1	G	
34	33	1998	31	F	2408003	MOSSORO	24	RN	1	Z	23.3
35	34	1998	21	M	2408003	MOSSORO	24	RN	1	Z	
36	35	1998	21	F	2408003	MOSSORO	24	RN	0		
37	36	1998	21	F	2408003	MOSSORO	24	RN	1	G	26.7

Figura 8.1: Pequena fotografia de uma planilha com os micro-dados do ENEM 1998. Fonte (Estudos e Pesquisas Educacionais Anísio Teixeira INEP, 2016).

8.1 O Cubo de Dados

Um metáfora usada para entender melhor as aplicações OLAP é a do Cubo de Dados. Usa-se um cubo porque é uma imagem 3D que pode ser entendida quando desenhada, e pode ser usada por algumas aplicações.

A ideia do cubo é estender para uma dimensão a mais o conceito de uma matriz de dados, para permitir visualizar melhor as operações OLAP.

Um cubo de dados mostra um conjunto de dados, equivalentes aos fatos de um Data Warehouse, de acordo com algumas dimensões. Por exemplo o cubo de dados OLAP da Figura 8.2 mostra a visão de dados sobre as vendas de uma rede de lojas de calçado. Em cada célula desse cubo está a informação desejada.

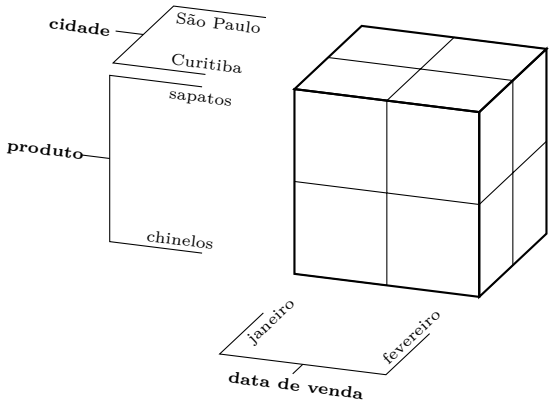


Figura 8.2: Cubo OLAP simplificado para rede de lojas de calçado.

Uma representação simples do cubo de dados OLAP é a de uma tabela com várias páginas. Assim, as linhas da tabela indicam a primeira dimensão, as colunas a segunda, e a terceira dimensão é indicada pela página.

8.2 OLAP com Pivot Table no Excel™

As planilhas eletrônicas normalmente possuem uma funcionalidade que permite fazer facilmente uma análise de dados OLAP. Essa funcionalidade de análise é conhecida como **pivot table**.

Por exemplo, a Figura 8.1 mostra uma imagem com um pedaço de uma planilha Excel™contendo uma única tabela com os micro-dados do ENEM de 1998(Estudos e Pesquisas Educacionais Anísio Teixeira INEP, 2016).

A partir dessa tabela foi criada outra aba, contendo uma *pivot table*, que foi configurada para contar as presenças. Para isso foi usado o comando Insert → Table → Pivot Table, o que faz aparecer o diálogo da Figura 8.3. Normalmente basta apertar Ok.

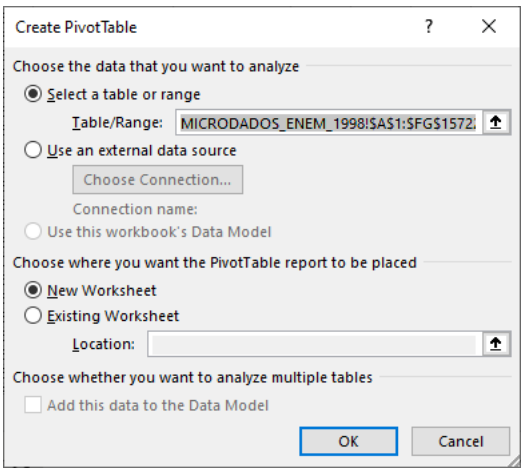


Figura 8.3: Diálogo do Pivot Table do Microsoft Excel™365.

A *pivot table* é controlada em duas áreas. Em uma, Pivot Table Field, detalhada na Figura 8.4 , é possível selecionar e configurar que campos serão mostrados de que forma, inclusive controlando fórmulas de cálculo.

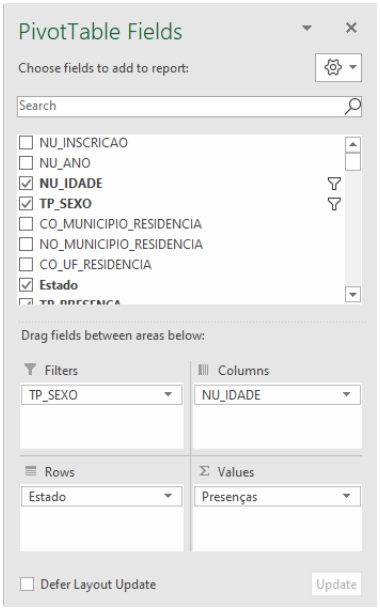
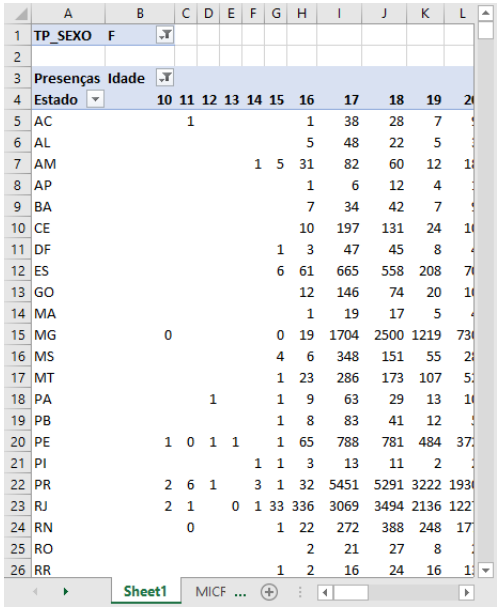


Figura 8.4: Nesse quadro são controladas que colunas aparecem em que posição na pivot table

Na outra, que parece uma planilha comum, Figura 8.5 , acontecem visualizações dinâmicas do que é selecionado e ainda podem ser editados alguns nomes e usados filtros.

Analisando as Figuras 8.4 e 8.5 é possível ver que foram selecionadas algumas colunas para aparecer na planilha. É possível definir três dimensões para o Cubo, *Filters*, que indica uma espécie de página, *Columns*, que indica as colunas da tabela e *Rows*, que



TP_SEXO	F											
Presenças	Idade											
Estado		10	11	12	13	14	15	16	17	18	19	2
AC			1					1	38	28	7	
AL								5	48	22	5	
AM						1	5	31	82	60	12	1
AP								1	6	12	4	
BA								7	34	42	7	
CE								10	197	131	24	1
DF							1	3	47	45	8	
ES							6	61	665	558	208	7
GO								12	146	74	20	1
MA								1	19	17	5	
MG		0					0	19	1704	2500	1219	73
MS							4	6	348	151	55	2
MT							1	23	286	173	107	5
PA				1			1	9	63	29	13	1
PB							1	8	83	41	12	
PE		1	0	1	1		1	65	788	781	484	37
PI						1	1	3	13	11	2	
PR		2	6	1		3	1	32	5451	5291	3222	193
RJ		2	1		0	1	33	336	3069	3494	2136	122
RN			0				1	22	272	388	248	17
RO								2	21	27	8	
RR							1	2	16	24	16	1

Figura 8.5: Na planilha, a pivot table vai se alterando dinamicamente para atender a configuração dada no quadro de controle.

indicam as linhas da tabela. Mais tarde no capítulo serão usadas hierarquias dentro das dimensões.

No quadro *Filters* foi escolhido o campo "TP_SEXO", que indica o sexo do candidato e só tem duas opções "F" ou "M". Para as linhas foi escolhido o campo "SG_UF_RESIDENCIA", que foi renomeado "para Estado". Para coluna foi escolhido o campo "NU_IDADE", que indica a idade do candidato. E para valor foi escolhido o campo "TP_PRESENÇA", também renomeado para "Presenças".

Clicando em "Presenças" é possível escolher como vai ser feito o cálculo desse campo e foi escolhido a forma *Sum*, já que a presença é marcado com o número 1 e a ausência com o 0. Isso é mostrado na Figura 8.6.

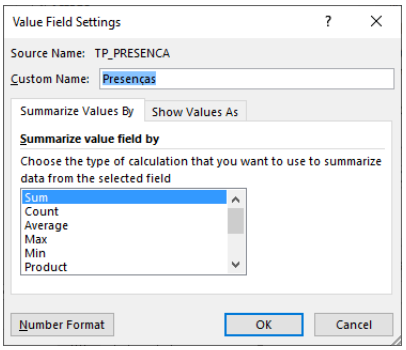


Figura 8.6: Escolhendo "Sum" como função de agregação de dados para a presença dos candidatos.

8.3 Slice and Dice

A primeira operação OLAP, **slice and dice** é bem fácil de entender e é bastante conhecida em outras áreas pelo nome de **filtro**.

Nessa operação simplesmente o usuário escolhe, dentro de uma visão que mostra muitos dados, uma coleção menor de dados para ver. O termo **slice** indica que ele fez essa escolha em apenas uma dimensão da consulta, cortando o cubo OLAP em uma fatia, e o termo **dice** indica que fez em várias, cortando essa fatia mais vezes e obtendo um cubo menor.

Usando o cubo da Figura 8.2, uma operação de *slice* poderia requisitar apenas os dados de São Paulo, resultando na Figura 8.7.

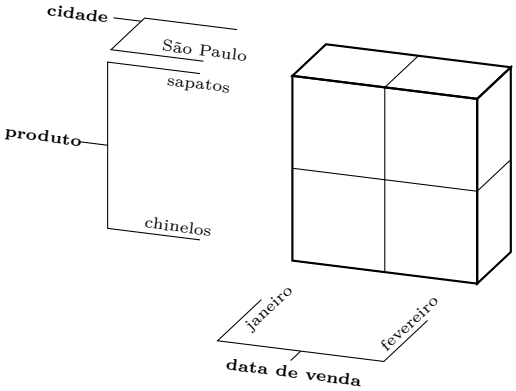


Figura 8.7: Efeito do uso de um filtro selecionando apenas os dados da cidade de São Paulo, realizando um *slice*.

E usando mais uma vez esse cubo, é possível selecionar apenas as vendas de janeiro, o que acaba resultando em um *dice*, como na Figura 8.8.

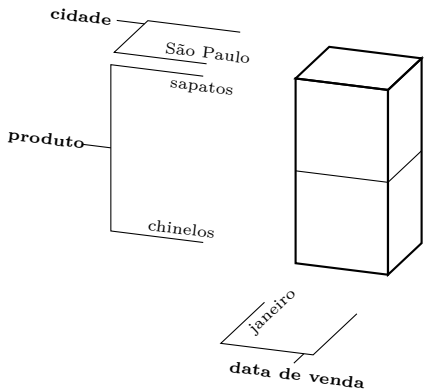


Figura 8.8: Aplicando mais de um filtro, obtemos um *dice*.

É possível observar que, em relação ao cubo original, a Figura 8.7 faz um corte, portanto é um *slice*, ou seja, uma fatia do cubo. Já na figura 8.8 mais um corte é aplicado e ficamos com um pedaço menor que uma fatia, um *dice*.

Por exemplo, a Tabela 8.1 apresenta os dados de todos os estados do Brasil, tornando difícil a comparação dos dados dos estados do Sul do Brasil, RS, PR e SC, entre eles. Uma operação de *slice* então corta essa fatia, por meio de filtros, e mostra os dados desejados, como na Tabela 8.2.

Tabela 8.1: Presenças por moradores dos estados no ENEM de 1998, Fonte: (Estudos e Pesquisas Educacionais Anísio Teixeira INEP, 2016).

Estado	**Presenças**
AC	368
AL	152
AM	355
AP	46
BA	170
CE	622
DF	199
ES	2650
GO	443
MA	80
MG	14679
MS	1070
MT	1192
PA	248
PB	269
PE	5665
PI	68
PR	47650
RJ	22296
RN	2584
RO	99
RR	546
RS	800
SC	1001
SE	289
SP	7433
TO	44
(blank)	4557
Total	115575

Tabela 8.2: Presenças por moradores dos estados no ENEM de 1998, Fonte: (Estudos e Pesquisas Educacionais Anísio Teixeira INEP, 2016).

Estado	Presenças
PR	47650
RS	800
SC	1001
Total	115575

8.3.1 Filtros (*Slice and Dice*) no Pivot Table do Excel™

Para usar os filtros do pivot table do Excel™é preciso apertar nas pequenas caixas com setas, ou filtros, que são vistas ao lado das palavras “Estado”, “Idade” e “TP_SEXO” na 8.5. No caso do aparecimento de um filtro, significa que há um filtro ativo. Nessa imagem já está sendo feito um filtro pelo sexo e um pela idade, logo caracterizando uma operação de *dice*.

Alterando os filtros, é possível obter outros resultados. Por exemplo, na Figura 8.9, são eliminadas as idades menores que 17 anos, resultando na tabela da Figura 8.10.

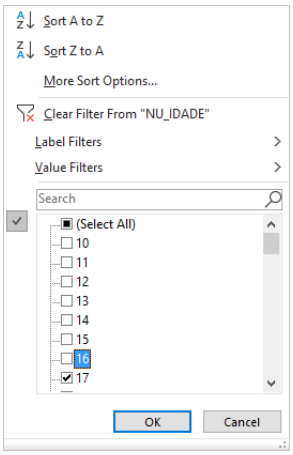
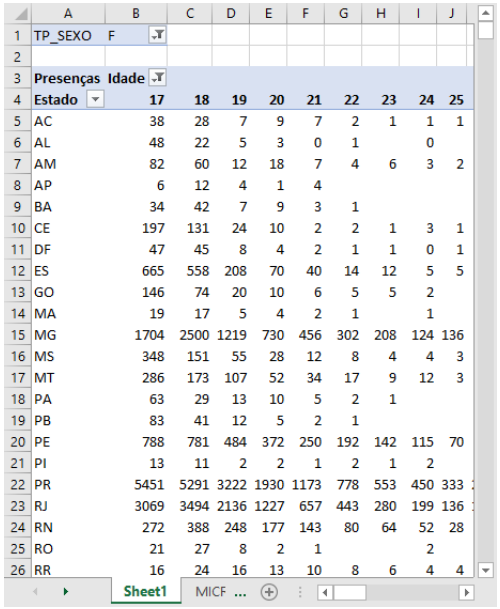


Figura 8.9: Usando o filtro para cortar algumas idades



	A	B	C	D	E	F	G	H	I	J
1	TP_SEXO	F								
2										
3	Presenças	Idade								
4	Estado	17	18	19	20	21	22	23	24	25
5	AC	38	28	7	9	7	2	1	1	1
6	AL	48	22	5	3	0	1		0	
7	AM	82	60	12	18	7	4	6	3	2
8	AP	6	12	4	1	4				
9	BA	34	42	7	9	3	1			
10	CE	197	131	24	10	2	2	1	3	1
11	DF	47	45	8	4	2	1	1	0	1
12	ES	665	558	208	70	40	14	12	5	5
13	GO	146	74	20	10	6	5	5	2	
14	MA	19	17	5	4	2	1		1	
15	MG	1704	2500	1219	730	456	302	208	124	136
16	MS	348	151	55	28	12	8	4	4	3
17	MT	286	173	107	52	34	17	9	12	3
18	PA	63	29	13	10	5	2	1		
19	PB	83	41	12	5	2	1			
20	PE	788	781	484	372	250	192	142	115	70
21	PI	13	11	2	2	1	2	1	2	
22	PR	5451	5291	3222	1930	1173	778	553	450	333
23	RJ	3069	3494	2136	1227	657	443	280	199	136
24	RN	272	388	248	177	143	80	64	52	28
25	RO	21	27	8	2	1			2	
26	RR	16	24	16	13	10	8	6	4	4

Sheet1 MICF ...

Figura 8.10: Tabela de presenças no ENEM 1998 por estado para maiores de 17 anos. Fonte (Estudos e Pesquisas Educacionais Anísio Teixeira INEP, 2016)

8.4 Drill-down e Roll-up

Quando estamos consultando uma base de dados é comum que uma pergunta possa ser feita em vários níveis de uma hierarquia que cobre o mesmo conceito.

Por exemplo, todas as perguntas a seguir querem obter informações sobre o valor total vendido, que é a informação importante, em um período de tempo, que é o conceito:

- Qual o valor total vendido por dia?
- Qual o valor total vendido por semana?
- Qual o valor total vendido por mês?
- Qual o valor total vendido por trimestre?
- Qual o valor total vendido por ano?

O mesmo pode ser visto em uma consulta sobre a quantidade de litros de combustível vendida por localidade:

- Qual a quantidade de litros de combustível vendida por posto de gasolina?
- Qual a quantidade de litros de combustível vendida por bairro?
- Qual a quantidade de litros de combustível vendida por cidade?
- Qual a quantidade de litros de combustível vendida por estado?
- Qual a quantidade de litros de combustível vendida por país?

Hieraquias como [dia - semana - mês - trimestre - ano] e [posto - bairro - cidade - estado - país] são muito comuns. Essas duas, a de intervalo de tempo e a de localidade ou endereço, inclusive, são as mais comuns e aparecem em quase todos data warehouses. Outras

hierarquias comuns são [produto - subcategoria - categoria], ou [setor - departamento - diretoria].

Essas hierarquias são muitas vezes desenhadas, como na Figura 8.11. Em especial, o desenho mostra que mês e semana devem ser vistos em paralelo, isto é, não é possível navegar de semana para mês nesse modelo.

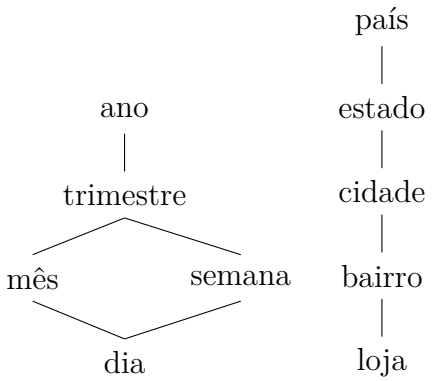


Figura 8.11: Hierarquias de conceitos. No caso, na hierarquia de períodos de tempo, temos dois caminhos paralelos.

Muitas vezes quando fazemos uma pergunta como as listadas acima, como "Qual o valor total vendido por trimestre?" ficamos insatisfeitos com o resultado e queremos mais detalhe. Por exemplo, queremos saber o valor total vendido por mês, em um determinado trimestre. Essa operação, onde pedimos mais detalhes sobre uma dimensão da pergunta, é conhecida como **drill down**, pois é como se estivéssemos "escavando mais profundamente" para entender o que está acontecendo.

Normalmente o *drill down* é feito em uma interface gráfica, onde o usuário clica em um dado para ver o detalhe. Por exemplo, clicando em um dado sobre um mês ele poderia ver os dados sobre cada dia do mês.

A ideia básica é demonstrada na Figura 8.12, escolhendo uma célula do cubo de OLAP e fazendo o *drill drown*, chegamos a um cubo mais detalhado.

Não é necessário todo um cubo para entender um *drill down*, basta olhar para uma tabela com dados agregados e procurar abrir esses dados de alguma forma prevista na interface do software.

A Tabela 8.1, por exemplo, mostra o número de presenças no ENEM de 1998, por estado de residência do candidato. Supondo que um usuário, vendo esse dado, queira saber de onde vieram os candidatos do Acre; em um sistema que suportasse o *drill down*, ele poderia clicar no Acre e receber uma visão da Tabela 8.3.

A operação inversa, que nos permite ter uma visão mais ampla, foi chamada inicialmente de **roll up**, e com o tempo ganhou também o nome de **drill up**[2].

[2]Que, convenhamos, não faz muito sentido

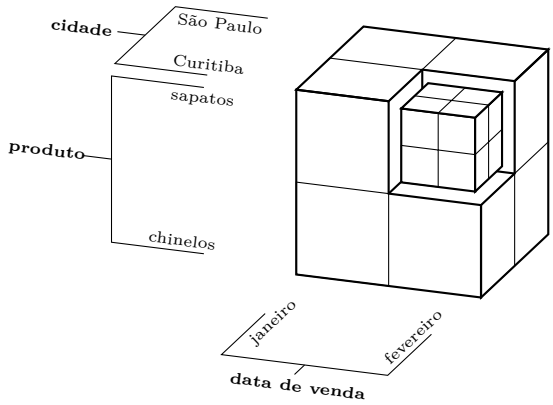


Figura 8.12: Fazendo um *drill down* se obtém um novo cubo mais detalhado.

Tabela 8.3: Presenças por moradores das cidades do Acre no ENEM de 1998. Fonte: (Estudos e Pesquisas Educacionais Anísio Teixeira INEP, 2016)

Estado	**Presenças**
ACRELANDIA	2
MANCIO LIMA	1
RIO BRANCO	365
Total AC	368

Na operação inversa é importante notar que a visão passa a um nível superior da hieraquia. Então, pedindo para fazer um roll-up quando vê a tabela de cidade do Acre (8.3), o usuário passaria a ver a tabela que inclui todos os estados, e o Acre como um deles (8.1).

8.4.1 Drill down e Roll Up

Nas linhas foi definida uma hierarquia [município - estado], como mostrado na Figura 8.13. A imagem ainda mostra, à direita, o painel de configuração da *pivot table*.

Clicando nos estados é possível abrir ou fechar o detalhe dos municípios, o equivalente à função de *drill down*. A Figura 8.14 mostra uma imagem da mesma pivot table com alguns estados fechados, ou seja, tendo sofrido um *roll up*.

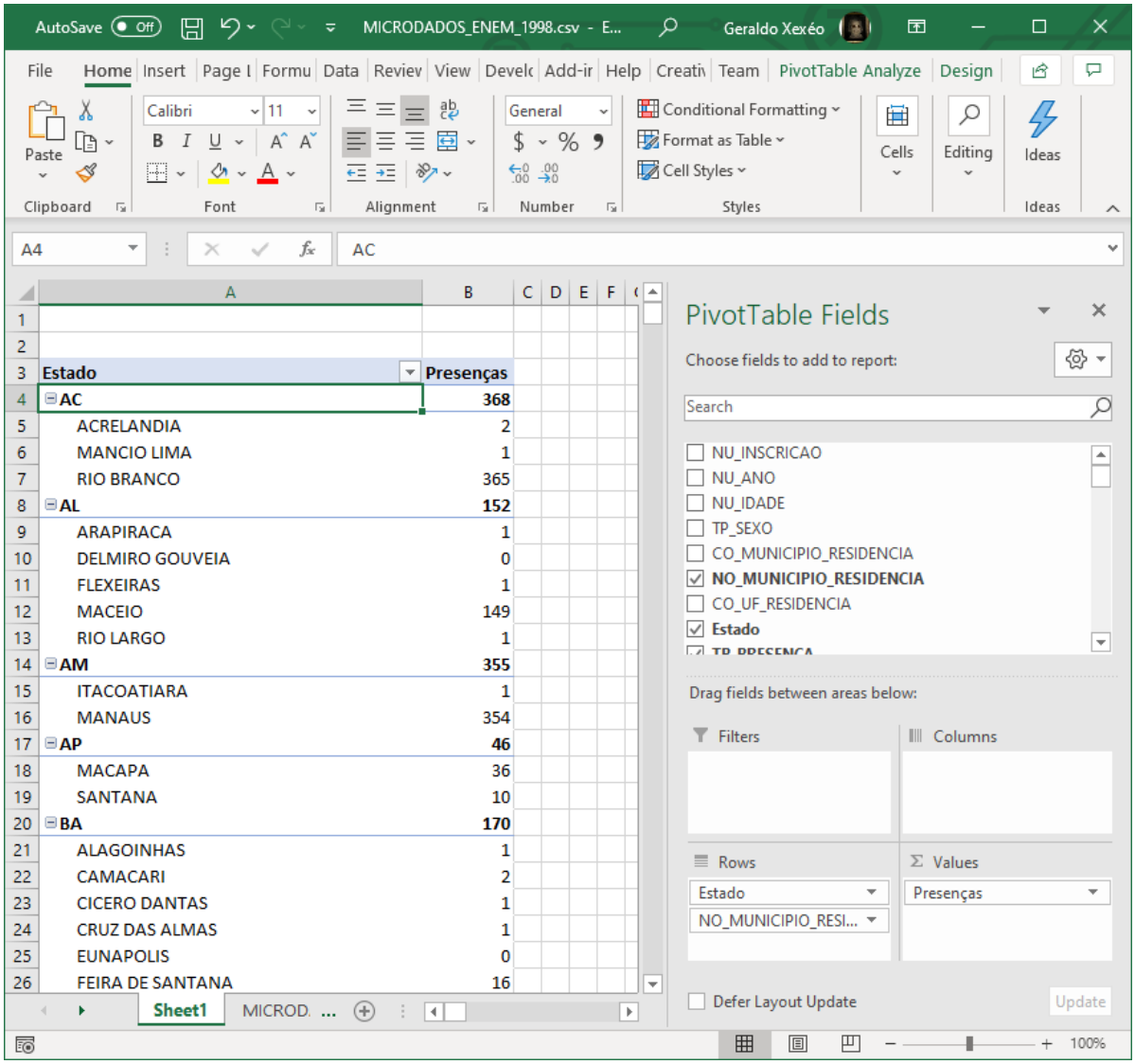


Figura 8.13: Pequena fotografia de uma planilha com uma pivot table verificando as presenças por cidade e estado do ENEM 1998. Fonte (Estudos e Pesquisas Educacionais Anísio Teixeira INEP, 2016).

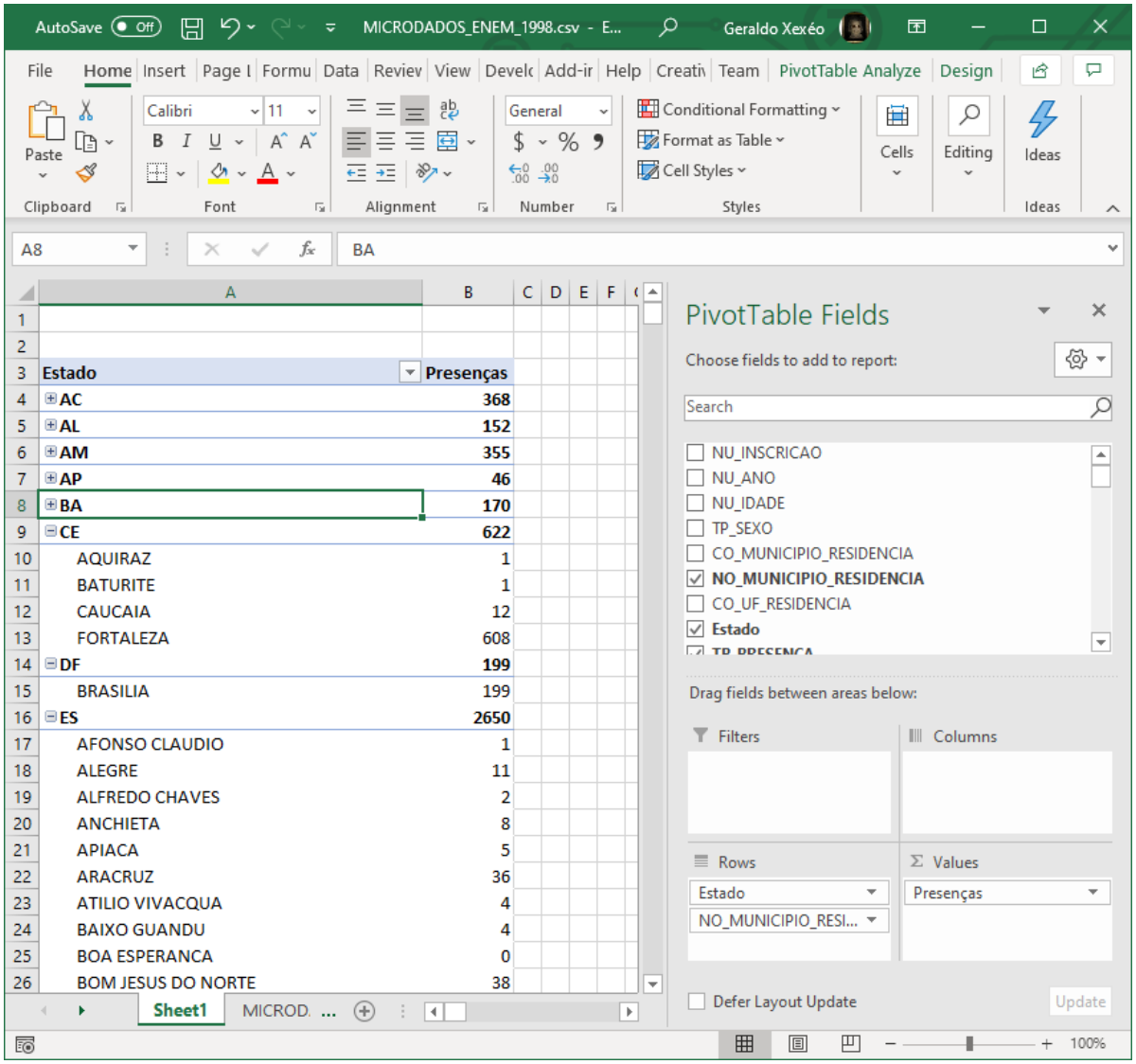


Figura 8.14: Pivot table com algumas informações suprimidas por meio de um *roll up* nos estados de Acre, Alagoas, Amazonas, Amapá e Bahia. Fonte (Estudos e Pesquisas Educacionais Anísio Teixeira INEP, 2016).

8.5 Pivot

A operação **pivot** é basicamente uma mudança de visualização, que reorienta o cubo. Seu efeito típico é trocar linhas por colunas, ou páginas por linhas.

Uma operação de *pivot* na Figura 8.2 poderia resultar na 8.15.

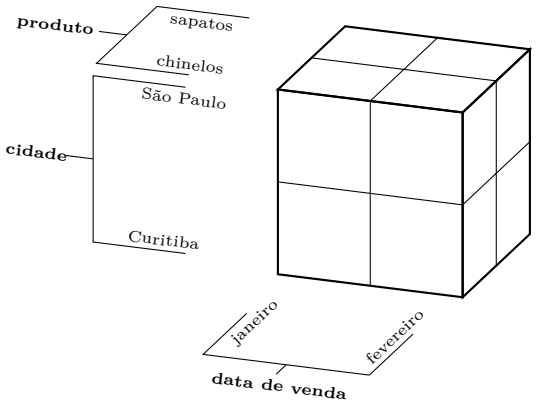


Figura 8.15: Operação de *pivot* no cubo OLAP troca a posição das dimensões. È necessário comparar com a Figura 8.2.

8.5.1 Pivot no Excel™

Já diz o nome que as *pivot table* no Excel™permitem facilmente fazer o *pivot*. Para isso basta um dado de linha para coluna e vice versa, com é demonstrado nas Figuras 8.16 e 8.17.

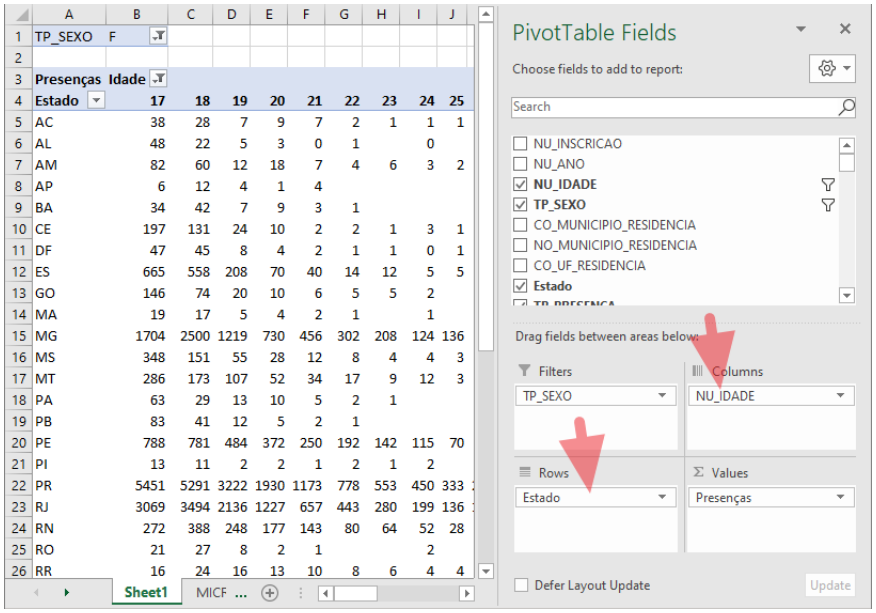


Figura 8.16: Posição dos dados antes da operação *pivot*.

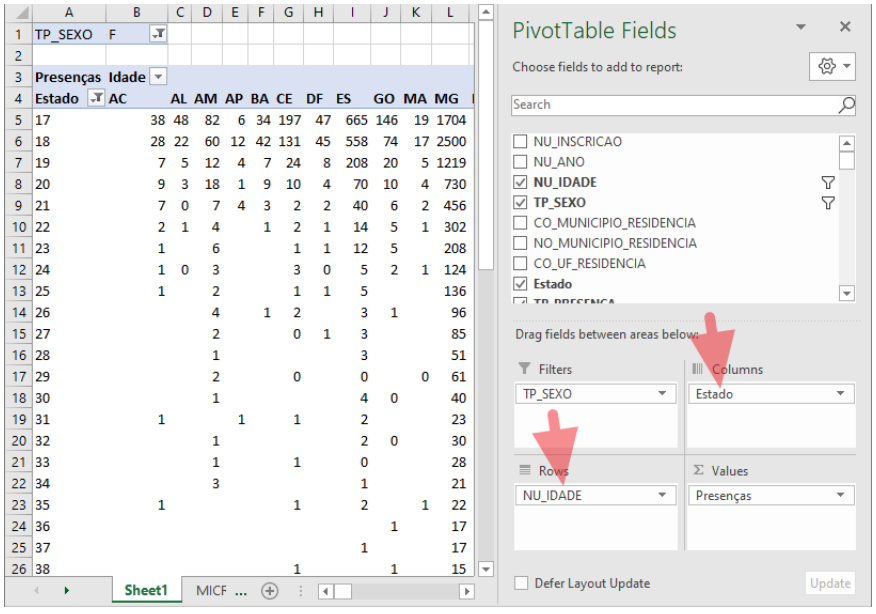


Figura 8.17: Após o *pivot*.

9 Modelagem Dimensional

Conteúdo

O objetivo da Modelagem Dimensional é fornecer ao usuários de um negócio uma visão intuitiva dos informações a que desejam ter acesso. O foco da Modelagem Dimensional é a simplicidade, ao contrário de outros modelos conceituais ou lógicos que buscam uma grande força de expressão ou flexibilidade no uso.

Outra função importante da Modelagem Dimensional é criar uma representação lógica que já direcione a organização das representações físicas possíveis para um alto desempenho de consulta.

Inicialmente o Modelo Dimensional era composto de uma tabela com uma chave composta com uma grande quantidade de dados, conhecida como Tabela Fato , que contém os dados numéricos que são manipulados nos relatórios e gráficos desejados, e um conjunto de tabelas que definem parâmetros que são usados em processos de agregação, projeção ou seleção, cada uma sendo uma Tabela Dimensão.

A ideia básica é que as tabelas dimensão fornecem as dimensões para a análise dos dados da tabela fato. Desta forma, cada tabela dimensão tem uma chave primária simples, que é referenciada pelas chaves externas da tabela fato. Isso faz com que a tabela fato seja cercada de outras tabelas, o que resulta em um modelo conhecido como Modelo Estrela.

O Modelo Dimensional, ou o Modelo Estrela, divide o mundo em dois grandes grupos de dados, Medidas e Contexto.

Medidas são dados capturados pelos sistemas de informação que executam os processos de negócio da organização. Normalmente são valores numéricos, sobre os quais podemos

fazer operações como somatório ou média. Números como o CPF e o Telefone de uma pessoa não são considerados medidas, e normalmente não são considerados números. As medidas definem os **fatos**(Kimball et al., 2008).

Contexto são os dados que mostram em que situação, momento, ou outros parâmetros os dados foram adquiridos. Normalmente são dados textuais ou datas, indicando informações como nomes, tipos de produto, datas de compra e venda, etc. O contexto é representado pelas dimensões.

As dimensões normalmente respondem as famosas questões do 5W2H: o que, onde, quando, quem, por que, como e por quanto(Corr e Stagnitto, 2012).

(Kimball et al., 2008) propõe que cada processo pode ser representado por um modelo dimensional, contendo uma tabela fato com as medidas numéricas levantadas ao longo de uma execução do proecesso, e tabelas que definem o contexto de cada medida. Na prática, várias tabelas fato podem ser levantadas para o mesmo processo, inclusive porque há escolhas de modelagem que são feitas, de acordo com a necessidade dos usuários, e também há uma quantidade de usuários e visões diferentes que um processo pode ter dentro de um sistema de data warehousing.

9.1 O Modelo Estrela

O Modelo Estrela é baseado em 4 conceitos(Kimball et al., 2008):

1. fatos,
2. dimensões,
3. atributos, e
4. relacionamentos.

9.1.1 A Tabela Fato

Uma tabela fato guada indicadores de desempenho gerados pela organização durante seu funcionamentos. Tipicamente elas possuem milhares ou milhões de linhas, ocupando grande parte da área de disco usada pelo sistema de data warehousing, chegando a 90% de todos os dados(Kimball et al., 2008).

Esses valores devem ser numéricos e aditivos, para serem analisados em tabelas e gráficos. Muitos deles representam dinheiro e quantidades.

Os tipos de fatos que podemos encontrar são(Kimball et al., 2008):

- **aditivos**, que são fatos a serem resumidos, podendo ser acumulados em qualquer dimensão; exemplos típicos são quantidade vendida e valor da venda;
- **semi-aditivos**, que só podem ser acumulados em algumas dimensões, como saldo em conta e quantidade em estoque, e

- **não aditivos**, que não podem ser resumidos, e são armazenados na dimensão, não sendo, na verdade, fatos, como o preço médio.

Na chave primária da tabela fato estão todas as chaves primárias das tabelas dimensão, representando uma ou mais relacionamentos 1:N com todas essas tabelas. Um exemplo de mais de um relacionamento seria um registro de compra e venda, que teria dois relacionamentos com a dimensão pessoa, uma indicando um vendedor, outra indicando um comprador.

É posssível que uma tabela fato não possua nenhum dado além das chaves estrangeiras(Kimball et al., 2008).

Bibliografia

American Heritage (2019). *The American Heritage Dictionary of the English Language*. URL: https://www.ahdictionary.com/ (acesso em 25/12/2019).

Bertini, C., S. Ceri e Shamkant B. Navathe (1992). *Conceptual Database Design*. The Benjamin/Cummings Publishing Company.

Celko, Joe (2005). *Joe Celko's SQL for smarties: advanced SQL programming*. Third. The Morgan Kaufmann series in data management systems. Los Altos, CA 94022, USA: Morgan Kaufmann Publishers, pp. xxviii + 808. ISBN: 0-12-369379-9 (paperback).

Chaitin, Gregore (mar. de 2006). "The Limits of Reason". *Scientific American*.

Chen, Peter (1990). *Modelagem de Dados: A abordagem entidade-relacionamento para projeto lógico*. São Paulo: Makron Books.

Claude E Shannon, Warren Weaver (1963). *The Mathematical Theory of Communication*. University of Illinois Press.

Coad, Peter, Jeff de Luca e Eric Lefebvre (1999). *Java Modeling Color with Uml: Enterprise Components and Process with Cdrom*. 1st. USA: Prentice Hall PTR. ISBN: 013011510X.

Codd, E. F. (jun. de 1970). "A relational model of data for large shared data banks". *Communications of the ACM* 13.6, pp. 377–387. DOI: 10.1145/362384.362685.

Corr, Lawrence e Jim Stagnitto (2012). *Agile Data Warhouse Design. Agile Data Warhouse Design: Collaborative Dimensional Modeling from Whiteboard to Star Schema. Collaborative Dimensional Modeling from Whiteboard to Star Schema*. Leeds, UK: Decision One Press.

Cougo, Paulo (1999). *Modelagem Conceitual e Projeto de Banco de Dados*. Rio de Janeiro: Campus.

Date, C. J. (2004). *An introduction to database systems*. Eighth. Boston, MA, USA: Pearson/Addison Wesley, pp. xxvii + 983 + 22. ISBN: 0-321-19784-4.

Elmasri, Ramez e Shamkant B. Navathe (2016). *Fundamentals of Database Systems*. 7th Edition. Boston: Pearson.

Estudos e Pesquisas Educacionais Anísio Teixeira INEP, Instituto Nacional de (5 de set. de 2016). *Microdados do Exame Nacional do Ensino Médio - Enem 1998*. http://portal.inep.gov.br/basica-levantamentos-acessar. Acessado em 25-10-2109. Instituto Nacional de Estudos e Pesquisas Educacionais Anísio Teixeira - INEP. URL: %5Curl%7B%20ftp://ftp.inep.gov.br/microdados/micro_enem1998.zip%7D (acesso em 26/10/2019).

Heuser, Carlos A. (2001). *Projeto de Banco de Dados*. Vol. 4. Série Livros Didáticos: Instituto de Informática da UFRGS. Porto Aleger: Editora Sagra Luzzatto.

Inmon, William H. (1992). *Building the Data Warehouse*. Wellesley, MA, USA: John Wiley & Sons. ISBN: 0-89435-404-3.

– (2005). *Building the Data Warehouse*. Fourth edition. Wiley Publishing, Inc. ISBN: 978-0-7645-9944-6.

ISO/IEC (jul. de 2004). *ISO/IEC 5218:2004 Information technology — Codes for the representation of human sexes*. ISO/IEC.

Kempe, Shannon e Paul Williams (ago. de 2012). *A Short History of Data Warehousing*. https://www.dataversity.net/a-short-history-of-data-warehousing. Acessado em 25/10/2019. URL: https://www.dataversity.net/a-short-history-of-data-warehousing/.

Kimball, Ralph et al. (2008). *The Data Warehouse Lifecycle Toolkit: Practical Techniques for Building Data Warehouse and Business Intelligence Systems*. 2nd. Wiley.

Microsoft (27 de out. de 2019). *Microsoft Excel 365*. [Computer Software]. M.

Motl, Jan (2019). *AdventureWorks*. URL: %5Curl%7Bhttps://docs.microsoft.com/en-us/sql/samples/adventureworks-install-configure?view=sql-server-ver15%7D.

NIST (1993). *Integration Definition for Information Modeling (IDEF1X)*. Federal Information Processing Standards Publication FIPS PUB 184. National Institute of Standards e Technology.

Nonaka, Ikujiro e Hirotaka Takeuchi (1995). *The Knowledge-Creating Company: How Japanese Companies Create the Dynamics of Innovation*. New York: Oxford University Press, xii, 284 p.

Oracle Corporation (2019a). *License for the Sakila Sample Database*. https://dev.mysql.com/doc/sakila/en/sakila-license.html. Accessed: 2019-10-23.

– (2 de nov. de 2019b). *MySQL 8.0 Reference Manual: Including MySQL NDB Cluster 8.0*. Acessado em 2019-11-02. Oracle Corporation. URL: %5Curl%7Bhttps://dev.mysql.com/doc/refman/8.0/en/%7D.

– (2019c). *Sakila Sample Database*. https://dev.mysql.com/doc/sakila/en/. Accessed: 2019-10-23.

Panko, Raymond R. (1998). "What We Know About Spreadsheet Errors". *Journal of End User Computing's* 10.2 (Spring 1998). Revised Mauy 2008, p. 15.21. URL: %5Curl%7Bhttp://panko.shidler.hawaii.edu/SSR/Mypapers/whatknow.htm%7D (acesso em 04/11/2019).

Parrochia, Daniel (2020). *Classification*. URL: https://www.iep.utm.edu/classifi/ (acesso em 21/01/2020).

Pressman, Roger e Bruce Maxim (2016). *Engenharia de Software-8ª Edição*. McGraw Hill Brasil.

Ralph Kimball, Margy Ross (12 de jul. de 2013). *The Data Warehouse Toolkit*. Wiley John + Sons. 608 pp. ISBN: 1118530802. URL: https://www.ebook.de/de/product/20197316/ralph_kimball_margy_ross_the_data_warehouse_toolkit.html.

Robertson, James e Suzzane Robertson (1998). *Complete Systems Analysis*. New York: Dorser House.

Rosen, Gideon (2018). "Abstract Objects". Em: *The Stanford Encyclopedia of Philosophy*. Ed. por Edward N. Zalta. Winter 2018. Metaphysics Research Lab, Stanford University.

Rowley, Jennifer (2006). "Where is the wisdom that we have lost in knowledge?" *Journal of Documentation* 62 (2), pp. 251–270.

– (2007). "The wisdom hierarchy: representations of the DIKW hierarchy." *Journal of Information Science* 33.2, pp. 163–180.

Scott, Dale (2016). *MySQL version of Northwind demo database*. https://github.com/dalers/mywind. Accessed: 2019-10-23.

Shlaer, Sally e Stephen J. Mellor (1999). *Object-Oriented Systems Analysis, Modelling the World in Data*.

Soto, Christiane (2019). *12 of the Biggest Spreadsheet Fails in History*. Acessado em 2019-11-04. Oracle Small-to-Medium Business Blog. URL: %5Curl%7B12%20of%20the%20Biggest%20Spreadsheet%20Fails%20in%20History%7D (acesso em 04/11/2019).

United States Code (jul. de 2002). *Sarbanes-Oxley Act of 2002, PL 107-204, 116 Stat 745*. Codified in Sections 11, 15, 18, 28, and 29 USC.

Wells, Dave e Eav Nahari (2019). *Modernizing the Legacy Data Warehousr: What, Why and How*. Acessado em 2019-11-3. Cloudera. URL: %5Curl%7Bhttps://www.slideshare.net/cloudera/modernizing-the-legacy-data-warehouse-what-why-and-how-12319%7D (acesso em 03/11/2019).

Yourdon, Ed (2006). *Just Enough Structured Analysis*. Ed Yourdon.

Zins, Chaim (fev. de 2007). "Conceptual Approaches for Defining Data, Information, and Knowledge: Research Articles". *J. Am. Soc. Inf. Sci. Technol.* 58.4, pp. 479–493. ISSN: 1532-2882.

– (2012). *Critical Delphi Research Methodology*. URL: http://www.success.co.il/critical-delphi.html (acesso em 31/12/2019).

Índice Remissivo