8 Business Intelligence

Business intelligence tornou-se um termo muito comum nos últimos anos. As ferramentas de banco de dados encontradas sob o título *business intelligence* incluem *data warehouse, on-line analytical processing* (OLAP) e *data mining*. As funcionalidades dessas ferramentas são complementares e inter-relacionadas. O *data warehouse* leva em consideração o armazenamento, a manutenção e a recuperação eficiente de dados históricos. O *OLAP* é um serviço que fornece respostas rápidas a consultas ocasionais realizadas no data warehouse. Os algoritmos de *data mining* encontram padrões dos dados e informam modelos ao usuário. Todas as três ferramentas estão relacionadas ao modo como os dados em um data warehouse são organizados logicamente, e o desempenho é altamente sensível às técnicas de projeto de banco de dados utilizadas [Barquin e Edelstein, 1997]. O objetivo por trás das tecnologias de business intelligence é fornecer informações úteis de apoio à decisão.

Os principais fornecedores de SGBD comercializam as ferramentas de data warehouse, OLAP e data mining como business intelligence. Este capítulo aborda cada uma dessas tecnologias. Examinamos de perto os requisitos de um data warehouse, seus componentes básicos e princípios de operação, os aspectos críticos em seu projeto e os elementos importantes do projeto lógico do banco de dados em seu ambiente. Depois, investigamos os elementos básicos do OLAP e data mining como técnicas de consulta especiais aplicadas ao data warehouse. Abordamos o data warehouse na Seção 8.1, OLAP na Seção 8.2 e data mining na Seção 8.3.

8.1 Data Warehouse

Um data warehouse é um grande repositório de dados históricos que podem ser integrados para apoiar decisões. O uso de um data warehouse é nitidamente diferente do uso dos sistemas operacionais*[1]. Os sistemas operacionais contêm os dados exigidos nas operações do dia-a-dia de uma organização. Esses dados operacionais costumam mudar rápida e constantemente. Os tamanhos de tabela nos sistemas operacionais são mantidos relativamente pequenos, eliminando-se dados antigos de tempos em tempos. O data warehouse, ao contrário, recebe periodicamente dados históricos em lotes e cresce com o tempo. O enorme tamanho dos data warehouses pode chegar a centenas de gigabytes, ou mesmo terabytes. Um problema que afeta o projeto de data warehouses é a necessidade de desenvolver consultas a grandes quantidades de dados e que retornem resultados rápidos . Os aspectos contrastantes entre os data warehouses e sistemas operacionais resultaram em uma abordagem de projeto diferenciada para o data warehouse.

8.1.1 Visão geral do data warehouse

Um data warehouse contém uma coleção de ferramentas de apoio à decisão, associadas a bancos de dados históricos com tamanhos significativos, permitindo que o usuário final tome decisões rápidas e seguras. O data warehouse surgiu da tecnologia de sistemas de apoio à decisão (DSS — Decision Support System) e dos sistemas de informação executivas (EIS — Executive Information Systems). Os DSSs são usados para analisar dados de bancos de dados, normalmente disponíveis de várias fontes, e criar relatórios. Os dados de relatórios não são de tempo crítico, no sentido de um sistema de tempo real, mas precisam estar atualizados para a tomada de decisão. Os EISs são como os DSSs, porém mais poderosos, mais fáceis de usar e mais específicos ao negócio. Os EISs foram criados para fornecer uma alternativa aos sistemas clássicos de processamento transacional (OLTP — on-line transaction processing), comuns à maioria dos sistemas de banco de dados disponíveis comercialmente. Os sistemas OLTP normalmente são usados para criar aplicações comuns, incluindo aquelas com prazos ou tempos de resposta de missão crítica. A Tabela 8.1 resume as diferenças básicas entre OLTP e sistemas data warehouse.

---

* *Nota dos Revisores Técnicos*: O termo sistemas operacionais é utilizado aqui em alusão aos sistemas transacionais que apóiam um negócio no nível operacional, tais como ERP, Workflow, Groupware, entre outros.

Tabela 8.1 Comparação entre OLTP e data warehouse

OLTP	*Data warehouse*
Orientado a transação	Orientado ao processo de negócios
Milhares de usuários	Poucos usuários (normalmente abaixo de 100)
Geralmente pequeno (MB até vários GB)	Grandes (de milhares de GB a vários TB)
Dados atuais	Dados históricos
Dados normalizados (muitas tabelas, poucas colunas por tabela)	Dados não normalizados (poucas tabelas, muitas colunas por tabela)
Atualizações contínuas	Atualizações em lote*
Consultas de simples a complexas	Normalmente, consultas muito complexas

* Atualmente existe um movimento no setor para o "active data warehouse", em que o warehouse recebe dados em atualizações contínuas. Veja uma discussão mais profunda na Seção 8.2.5.

A arquitetura básica para um ambiente de data warehouse aparece na Figura 8.1. O diagrama mostra que o data warehouse é alimentado por uma série de bancos de dados fonte, possivelmente de diferentes localizações geográficas. Cada banco de dados fonte serve às suas próprias aplicações, e o data warehouse serve a um DSS/EIS atendendo as suas solicitações de informação. Cada banco de dados do sistema fonte precisa ser reconciliado com o modelo de dados do data warehouse; isso é feito durante o processo de extração dos dados exigidos do sistema de banco de dados fonte, transformando os dados do sistema fonte para o data warehouse e carregando os dados no data warehouse [Cataldo, 1997].

Requisitos básicos para o data warehouse

Agora, vamos examinar os principais requisitos e princípios que orientam o projeto de data warehouses (DWs) [Simon, 1995; Barquin e Edelstein, 1997; Chaudhuri e Dayal, 1997; Gray e Watson, 1998]:

1. DWs são organizados de acordo com as áreas de interesse. As áreas de interesse são semelhantes ao conceito de áreas funcionais, como vendas, gerenciamento de projetos ou funcionários, conforme discutimos no contexto do agrupamento do diagrama ER na Seção 4.5. Cada área de interesse tem seu próprio esquema conceitual e pode ser representada usando uma ou mais entidades no modelo de dados ER ou por uma ou mais classes de objeto no modelo de dados orientado a objeto. As áreas de interes-

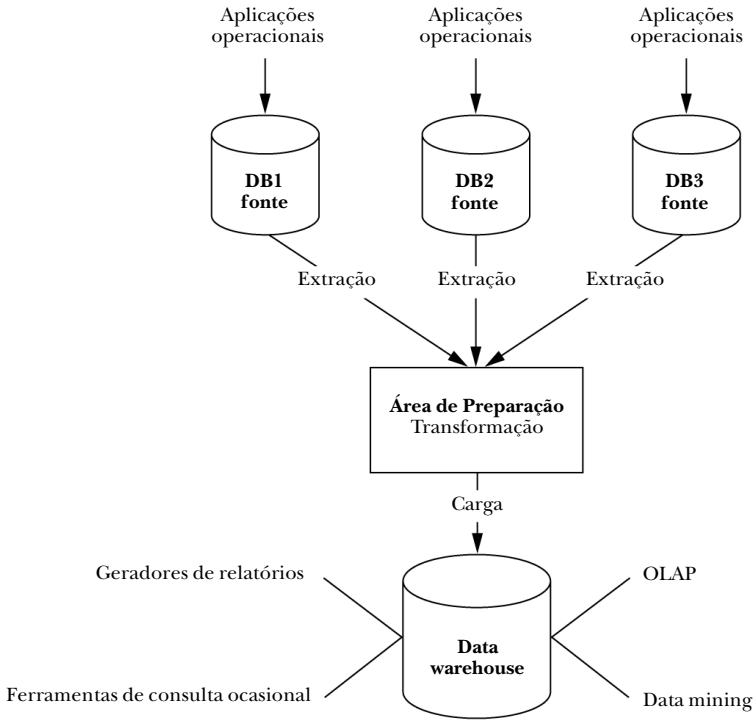


Figura 8.1 Arquitetura básica do data warehouse.

se normalmente são independentes das transações individuais que envolvem a criação ou manipulação de dados. Os repositórios de metadados são necessários para descrever os bancos de dados fontes, objetos DW e maneiras de transformar os dados fontes para o DW.

2. Os DWs devem ter alguma capacidade de integração. Uma representação comum dos dados deverá ser projetada de modo que todas as diferentes representações individuais possam ser mapeadas. Isso é particularmente útil se o warehouse for implementado como um multibanco de dados ou banco de dados federado.
3. Os dados são considerados não voláteis e devem ser carregados em massa. A extração de dados dos bancos de dados atuais para o DW exige que se tome uma decisão quanto a extrair os dados usando técnicas padrão de banco de dados relacional (BDR) no nível de linha ou coluna, ou técnicas especializadas para extração em massa. Ferramentas de limpeza de dados são necessárias para manter a qualidade dos dados - por exemplo, detec-

tar dados em falta, dados inconsistentes, homônimos, sinônimos e dados com diferentes unidades. Ferramentas de migração de dados, tratamento de dados e auditoria de dados cuidam de problemas especializados em limpeza e transformação de dados. Essas ferramentas são semelhantes àquelas utilizadas para a integração convencional de esquemas (view) de bancos de dados relacionais. Utilitários de carga obtêm dados limpos e os carregam no DW, usando técnicas de processamento em lote. As técnicas de atualização propagam as alterações nos dados fonte para os dados básicos e dados derivados do DW. A decisão de quando e como atualizar é realizada pelo administrador do DW e depende das necessidades do usuário (por exemplo, necessidades do OLAP) e do tráfego existente no DW.

4. Os dados tendem a existir em vários níveis de granularidade. Mais importante, os dados costumam ser de natureza histórica, com grande potencial de variarem com o passar do tempo. Entretanto, em geral, a granularidade pode variar de acordo com as diferentes dimensões e não apenas com o tempo; podem variar por região geográfica, tipo de produto manufaturado ou vendido, tipo de loja e assim por diante. O enorme tamanho dos bancos de dados é um problema importante no projeto e implementação de DWs, especialmente para certas consultas, atualizações e backups seqüenciais. Isso faz com que uma decisão crítica tenha de ser tomada entre usar um banco de dados relacional (BDR) e usar um banco de dados multidimensional (BDM) para implementar um DW.
5. O DW deverá ser suficientemente flexível para atender rapidamente a necessidade por constantes mudanças. As definições de dados (esquemas) precisam ser suficientemente amplas para antecipar acréscimos de novos tipos de dados. Para a necessidade de recuperar dados que estão sofrendo rápidas mudanças, os tipos de dados e os níveis de granularidade que serão realmente implementados precisam ser escolhidos cuidadosamente.
6. O DW deverá ter a capacidade de reescrever a história, ou seja, permitir análises hipotéticas ("o que acontece se"). O DW deverá permitir que o administrador altere temporariamente os dados históricos com o objetivo de realizar análises hipotéticas. Quando a análise terminar, os dados precisam ser revertidos corretamente. Esta condição considera que os dados estão no nível de granularidade apropriado em primeiro lugar.
7. Deverá ser selecionada uma interface de usuário útil para o DW. As principais escolhas hoje são a SQL, visões multidimensionais de dados relacionais, ou uma interface de usuário de uso especial. A linguagem da interface de usuário precisa ter ferramentas para recuperar, formatar e analisar dados.

8. Os dados devem estar centralizados ou distribuídos fisicamente. O DW deverá ter a capacidade de lidar com dados distribuídos em uma rede. Essa necessidade se tornará mais crítica à medida que o uso de DWs cresce e as origens dos dados se expandem.

O ciclo de vida dos data warehouses

Livros inteiros têm sido escritos a respeito de partes selecionadas do ciclo de vida do data warehouse. Nossa finalidade nesta seção é apresentar alguns fundamentos e sabores do data warehouse. Incentivamos aos que desejam prosseguir na área de data warehouse a continuar seus estudos através de outros livros dedicados especificamente ao data warehouse. Kimball e Ross [1998, 2002] possuem uma série de livros excelentes e que abordam detalhes das atividades de data warehouse.

A Figura 8.2 esboça as atividades do ciclo de vida do data warehouse, com base na Figura 16.1 de Kimball e Ross [2002]. O ciclo de vida começa com um diálogo para determinar o plano de projeto e as necessidades de negócios. Quando o plano e as necessidades são alinhados, o projeto e a implementação podem prosseguir. O processo se ramifica em três linhas de execução independentes e se juntam antes da implantação (ver Figura 8.2). Questões de plataforma são abordadas em um ramo, incluindo o projeto técnico arquitetônico, seguido pela seleção e instalação de produto. As questões de dados são abordadas no segundo ramo, incluindo a modelagem dimensional e depois o projeto físico, seguido pelo projeto e desenvolvimento da preparação de dados. As necessidades analíticas especiais dos usuários são atendidas no terceiro ramo, incluindo a especificação da aplicação analítica, seguida pelo desenvolvimento da aplicação analítica. Esses três ramos de execução se juntam antes da implementação. A implementação é seguida pela manutenção e pelo crescimento, e as mudanças nas necessidades precisam ser detectadas. Se forem necessários ajustes, o ciclo se repetirá. Se o sistema se tornar inativo, então o ciclo de vida termina.

O restante da nossa seção de data warehouse aborda a atividade de modelagem dimensional. Um material mais abrangente poderá ser encontrado em Kimball e Ross [1998, 2002] e Kimball e Caserta [2004].

8.1.2 Projeto lógico

Discutimos o projeto lógico de data warehouses nesta seção; as questões de projeto físico são abordadas no volume dois. O projeto lógico de data warehouses

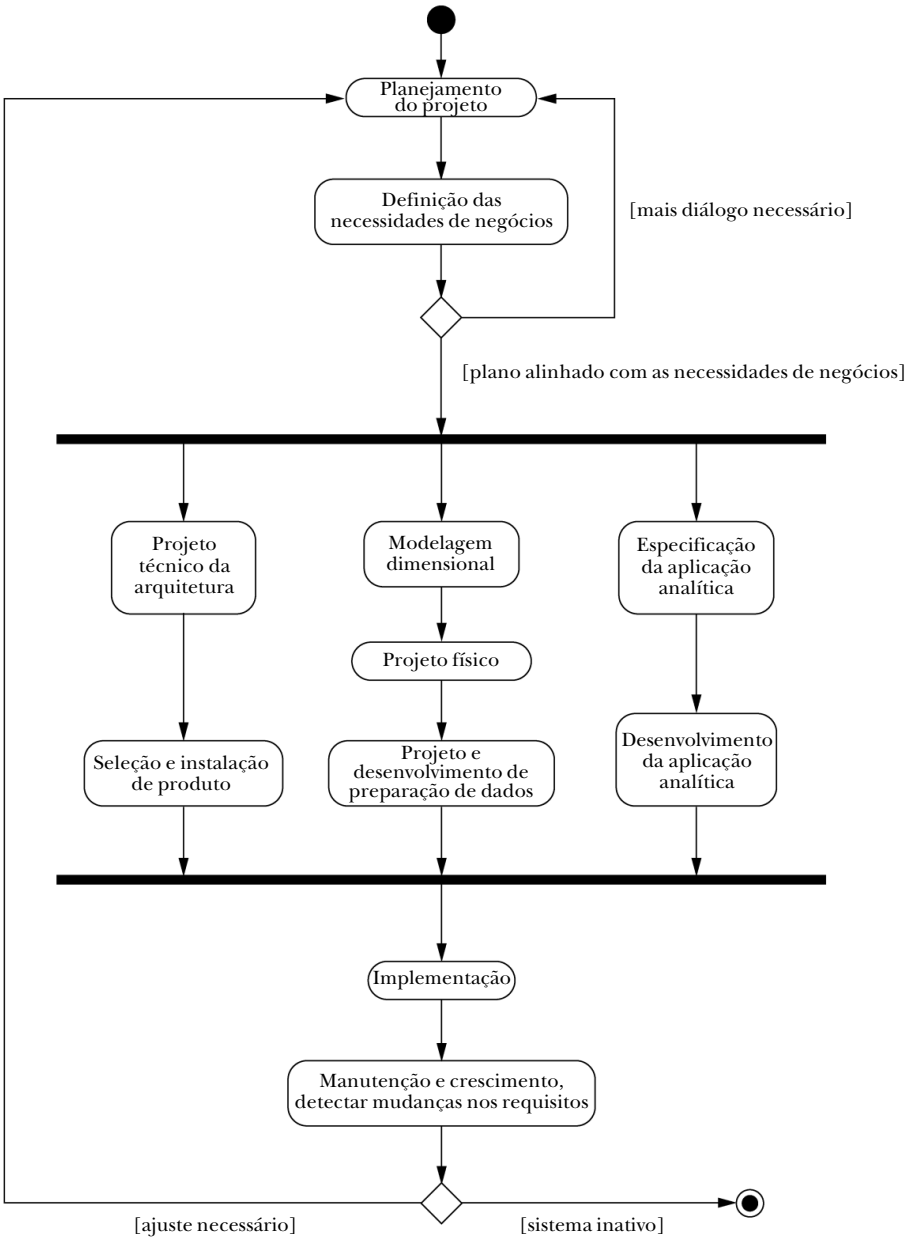


Figura 8.2 Ciclo de vida do data warehouse (baseado em Kimball e Ross [2002], Figura 16.1).

é definido pela abordagem de modelagem de dados dimensional. Vamos tratar dos tipos de esquema normalmente encontrados na modelagem dimensional, incluindo o *star schema* (esquema estrela) e o esquema *snowflake* (floco de neve). Esboçamos o processo de projeto dimensional conforme a metodologia descrita por Kimball e Ross [2002]. Depois, percorremos um exemplo, abordando alguns dos conceitos cruciais da modelagem de dados dimensional.

Modelagem de dados dimensional

A abordagem de modelagem dimensional é muito diferente da abordagem de normalização em geral seguida quando se projeta um banco de dados para as operações do dia-a-dia. O contexto do data warehouse exige uma abordagem diferente para atender às necessidades do usuário. A necessidade de modelagem dimensional será mais bem discutida enquanto prosseguirmos. Se você ainda não foi exposto ao data warehouse, esteja preparado para alguns paradigmas novos.

O Star Schema

Os data warehouses normalmente são organizados em uma grande *tabela de fatos* central e muitas *tabelas de dimensão* menores. Essa configuração é chamada de *star schema*; um exemplo é apresentado na Figura 8.3. A tabela de fatos é composta de dois tipos de atributos: *atributos de dimensão* e *atributos de medidas*. Os atributos de dimensão na Figura 8.3 são CustID, ShipDateID, BindID e JobId. A maioria dos atributos de dimensão possui relacionamentos de chave estrangeira/chave primária com as tabelas de dimensão. As tabelas de dimensão na Figura 8.3 são Customer (Cliente), Ship Calendar (Calendário de Embarque) e Bind Style (Estilo da Embalagem). Ocasionalmente, um atributo de dimensão existe sem uma tabela de dimensão relacionada. Kimball e Ross referem-se a elas como *dimensões degeneradas*. O atributo JobId na Figura 8.3 é uma dimensão de degeneração (veja mais sobre isso em breve). Indicamos os atributos de dimensão que atuam como chaves estrangeiras usando o estereótipo «fk». As chaves primárias das tabelas de dimensão são indicadas com o estereótipo «pk». Quaisquer dimensões degeneradas na tabela de fatos são indicadas com o estereótipo «dd». A tabela de fatos também contém medidas, que contêm valores a serem agregados quando as consultas agrupam linhas. As medidas na Figura 8.3 são Cost (preço de custo) e Sell (preço de venda).

As consultas no star schema normalmente utilizam atributos nas tabelas de dimensão para selecionar as linhas pertinentes da tabela de fatos. Por exemplo, o usuário pode querer ver o preço de custo e de venda de todas as tarefas

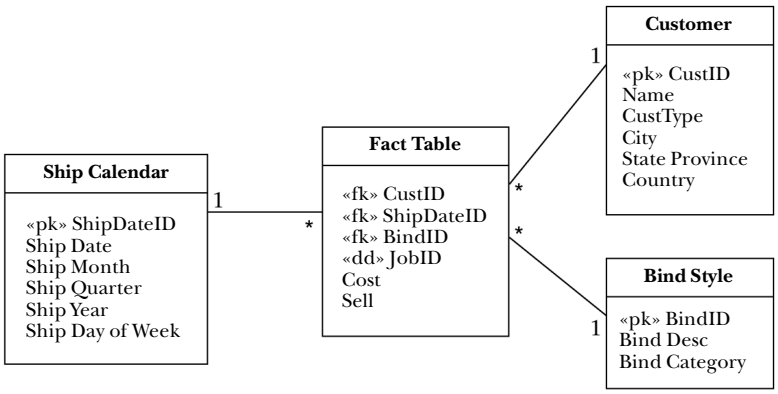


Figura 8.3 Exemplo de um star schema de um data warehouse.

em que Ship Month (Mês de Embarque) seja janeiro de 2005. Os atributos da tabela de dimensão normalmente também são usados para agrupar as linhas de maneiras úteis quando se exploram informações de resumo. Por exemplo, o usuário pode querer ver o custo total e o preço de venda de cada Ship Month no Ship Year (Ano de Embarque) de 2005. Observe que as tabelas de dimensão podem permitir diferentes *níveis* de detalhe para o usuário examinar. Por exemplo, o esquema da Figura 8.3 permite que as linhas da tabela de fato sejam agrupadas por Ship Date (Data de Embarque), Month (Mês), Quarter (Quadrimestre) ou Year (Ano). Esses níveis de dimensão formam uma *hierarquia*. Há também uma segunda hierarquia na dimensão Ship Calendar que permite que o usuário agrupe as linhas da tabela de fatos por Ship Day of Week (dia da semana). O usuário pode subir ou descer em uma hierarquia ao explorar os dados. Descer na hierarquia para examinar dados mais detalhados é uma operação chamada *drill-down*. Subir em uma hierarquia para resumir detalhes é uma operação *roll-up*.

Juntos, os atributos de dimensão compõem uma chave candidata da tabela de fatos. O nível de detalhe definido pelos atributos de dimensão é a *granularidade* da tabela de fatos. Ao projetar uma tabela de fatos, a granularidade dever ser o nível mais detalhado disponível que qualquer usuário desejaria examinar. Esse requisito às vezes significa que uma dimensão degenerada, como JobId na Figura 8.3, precisa ser incluída. O JobId nesse star schema não é usado para selecionar ou agrupar linhas, de modo que não existe tabela de dimensão relacionada. A finalidade do atributo JobId é distinguir as linhas no nível correto de granularidade. Sem o atributo JobId, a tabe-

la de fatos agruparia tarefas semelhantes, proibindo o usuário de examinar os preços de custo e de venda de tarefas individuais.

A normalização não é o princípio orientador no projeto do data warehouse. A finalidade do data warehouse é fornecer respostas rápidas a consultas envolvendo um grande conjunto de dados históricos. A organização do star schema facilita a resposta rápida a consultas no contexto do data warehouse. Os principais dados detalhados são centralizados na tabela de fatos. As informações dimensionais e hierarquias são mantidas nas tabelas de dimensão, com uma única junção com a tabela de fatos. Os níveis hierárquicos de dados contidos nas tabelas de dimensão da Figura 8.3 violam a 3FN, mas essas violações dos princípios de normalização são justificadas. O processo de normalização desmembraria cada tabela de dimensão na Figura 8.3 em várias tabelas. O esquema normalizado resultante exigiria maior processamento de junção na maioria das consultas. As tabelas de dimensão são pequenas em comparação a tabela de fatos, e normalmente mudam lentamente. O núcleo das operações no data warehouse são operações de leitura. Os benefícios da normalização são pequenos quando a maioria das operações é apenas de leitura. Os

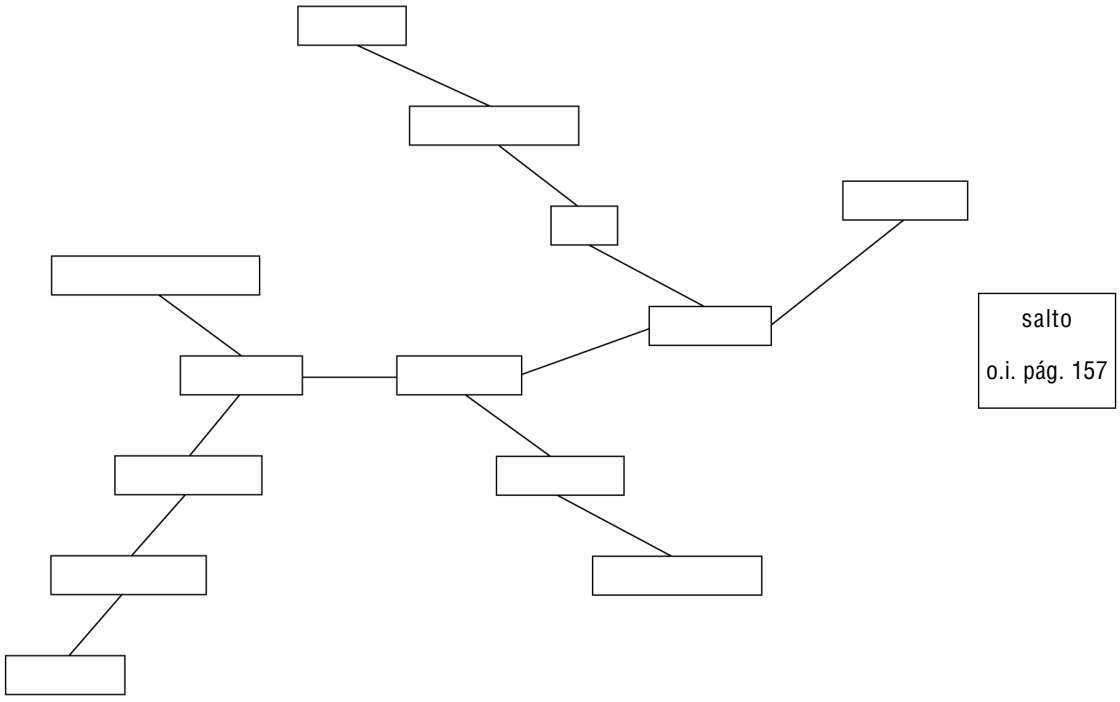


Figura 8.4 Exemplo de um esquema snowflake de um data warehouse.

benefícios de minimizar operações de junção ultrapassam os benefícios da normalização no contexto do data warehouse. As diferenças sinalizadas entre o ambiente de data warehouse e o ambiente do sistema operacional levam a abordagens de projeto distintas. A modelagem dimensional é o princípio básico no projeto do data warehouse.

Esquema Snowflake

A literatura de data warehouse normalmente se refere a uma variação do star schema conhecida como *esquema snowflake*. A normalização das tabelas de dimensão em um star schema leva a um esquema snowflake. A Figura 8.4 mostra o esquema snowflake semelhante ao star schema da Figura 8.3. Observe que cada nível hierárquico se torna sua própria tabela. O esquema snowflake está caindo em desgraça. Kimball e Ross preferem fortemente o star schema, devido à sua velocidade e simplicidade. O star schema não apenas gera uma resposta de consulta mais rápida, mas também é mais fácil para o usuário entender quando cria consultas. Incluímos o esquema snowflake aqui apenas para termos a visão geral.

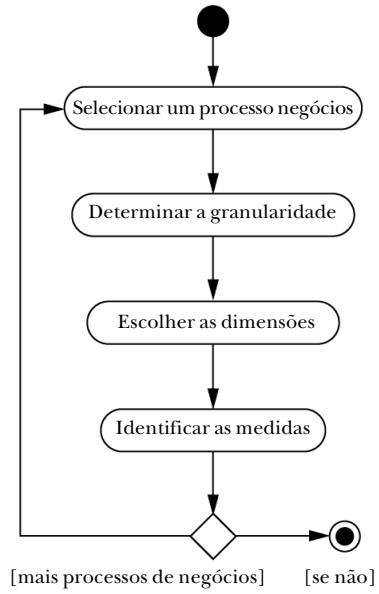


Figura 8.5 Processo em quatro etapas do projeto dimensional [Kimball e Ross, 2002].

Processo de projeto dimensional

Reunimos as quatro etapas do processo de projeto dimensional promovido por Kimball e Ross. A Figura 8.5 esboça as atividades do processo em quatro etapas.

Exemplo de modelagem dimensional

Parabéns, você agora é o proprietário da ACME Data Mart Company! Sua empresa cria data warehouses. Você presta consultorias a outras empresas, projeta e implanta data warehouses para atender a suas necessidades e oferece suporte em seus esforços.

Seu primeiro cliente é a XYZ Widget, Inc. XYZ Widget é uma empresa de manufatura com sistemas de informação prontos. São sistemas operacionais que rastreiam o estado atual e recente dos diversos processos de negócios. Os registros mais antigos, que não são mais necessários para a operação da fábrica, são eliminados. Isso mantém os sistemas operacionais funcionando de modo eficiente.

A XYZ Widget tem dez anos e está crescendo rapidamente. A gerência observa que as informações são valiosas. O gerente de tecnologia da informação vem salvando dados antes que eles fossem eliminados do sistema operacional. Existem dezenas de milhões de registros históricos, mas não há um modo fácil de acessar os dados de forma significativa. A ACME Data Mart foi chamada para projetar e montar um DSS acessando esses dados históricos.

As discussões com a XYZ Widget começam. Existem muitas perguntas cujas respostas precisam analisar os dados históricos. Você começa fazendo uma lista daquilo que a XYZ deseja saber.

Lista de desejos da XYZ Widget Company

1. Quais são as tendências dos diversos produtos em termos de valores de vendas, volume unitário e margem de lucro?
2. Para os produtos que não são lucrativos, podemos nos aprofundar e determinar por que não são lucrativos?
3. Com que precisão nossos custos estimados correspondem aos nossos custos reais?
4. Quando mudamos nossos cálculos de estimativa, como as vendas e a lucratividade são afetadas?
5. Quais são as tendências na porcentagem de tarefas que são entregues em tempo?

6. Quais são as tendências na produtividade por departamento, para cada máquina e para cada funcionário?
7. Quais são as tendências para atender as datas agendadas de cada departamento e de cada máquina?
8. Qual foi a eficiência do upgrade na máquina 123?
9. Quais clientes trazem os trabalhos mais lucrativos?
10. Como nossos descontos promocionais para compra em massa afetam as vendas e a lucratividade?

Examinando a lista de desejos, você começa a selecionar os processos de negócios envolvidos. A lista a seguir é suficiente para satisfazer aos itens na lista de desejos.

Processos de negócios

1. Estimativa
2. Agendamento
3. Rastreamento da produtividade
4. Custos do trabalho

Esses quatro processos de negócios estão interligados na XYZ Widget Company. Vamos examinar rapidamente os processos de negócios e a organização da informação nos sistemas operacionais, para termos uma idéia de qual informação está disponível para análise. Para cada processo de negócio, vamos projetar um star schema para armazenar os dados.

O processo de estimativa começa entrando-se com as especificações do produto. O tipo de produto determina quais máquinas são usadas para manufaturá-lo. O software de estimativa, então, calcula o tempo estimado em cada máquina usada para produzir esse tipo específico de produto. Cada máquina é modelada com um tempo de preparação padrão e velocidade de execução. Se determinado tipo de produto é difícil de processar em determinada máquina, os tempos são ajustados de acordo. Cada máquina possui um valor hora. O tempo estimado é multiplicado pelo valor hora para obter o custo de mão-de-obra. Cada estimativa armazena especificações de produto, um desmembramento dos custos de manufatura, o acréscimo e o desconto aplicado (se houver) e o preço. A cotação é enviada ao cliente. Se o cliente aceitar a cotação, então ela será associada ao número da tarefa, as especificações serão impressas como um ticket de tarefa, e este passará para o agendamento.

Precisamos determinar a granularidade antes de projetar o esquema do data mart de estimativa. A granularidade deve ser no nível mais detalhado, para dar maior flexibilidade às operações de detalhamento quando os usuários estiverem explorando os dados. O nível de maior granularidade no processo de estimativa é o detalhe de estimativa. Cada registro de detalhe de estimativa especifica informações de um centro de custo individual para determinada estimativa. Essa é a granularidade mais fina dos dados de estimativa no sistema operacional, e esse nível de detalhe também é potencialmente valioso para os usuários do data warehouse.

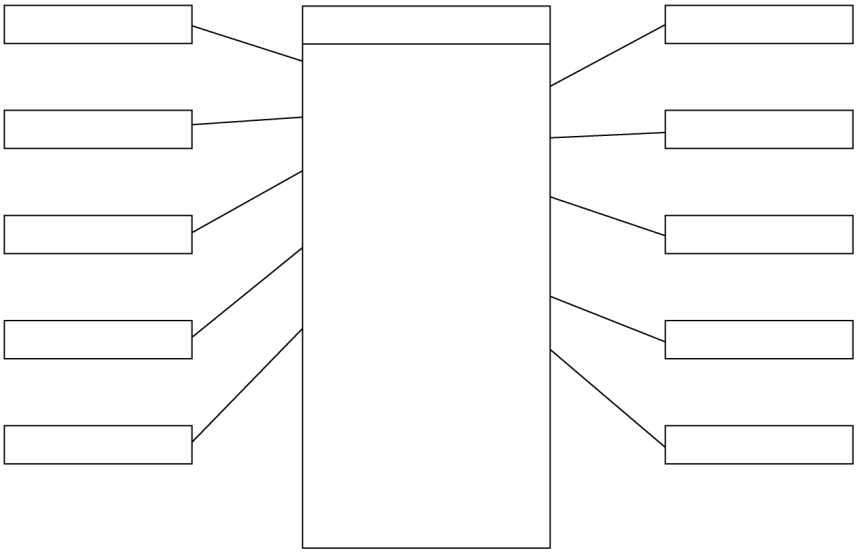
A próxima etapa do projeto é determinar as dimensões. Examinando os detalhes de estimativa, vemos que os atributos associados são as especificações de tarefa, o número estimado e data, o número da tarefa e data de vitória se a estimativa se tornar uma tarefa, o cliente, a promoção, o centro de custo, a quantidade de produtos, horas estimadas, valor hora, custo estimado, acréscimo, desconto e preço. As dimensões são aqueles atributos que os usuários desejam agrupar quando exploram os dados. Os usuários estão interessados em agrupar pelas diversas especificações de tarefa e pelo centro de custo. Os usuários também precisam ser capazes de agrupar por intervalos de data. A data estimada e a data de vitória são de interesse. O agrupamento por cliente e promoção também é de interesse dos usuários. Elas se tornam as dimensões do star schema para o processo de estimativa.

Em seguida, identificamos as medidas. As medidas são as colunas que contêm valores a serem agregados quando as linhas são agrupadas. As medidas no processo de estimativa são horas estimadas, valor hora, custo estimado, acréscimo, desconto e preço.

O star schema resultante da análise do processo de estimativa aparece na Figura 8.6. Existem cinco qualidades de produtos de nosso interesse: forma, cor, textura, densidade e tamanho. Por exemplo, um determinado produto poderia ser leve, macio, vermelho, redondo e médio. Os números de estimativas e tarefas são incluídos como dimensões degeneradas. O restante das dimensões e medidas é esboçado conforme nos dois parágrafos anteriores.

Os valores de dimensão são categóricos por natureza. Por exemplo, um determinado produto poderia ter uma densidade leve ou pesada. Os valores para a dimensão de tamanho são: pequeno, médio e grande. As medidas costumam ser numéricas, pois normalmente são agregadas usando funções como soma ou média.

As tabelas de dimensão devem possuir quaisquer hierarquias que possam ser úteis para análise. Por exemplo, produtos são oferecidos em muitas cores.

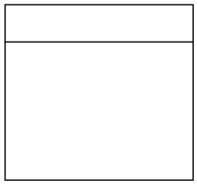


salto
o.i. pág. 160

Figura 8.6 Star schema para o processo de estimativa.

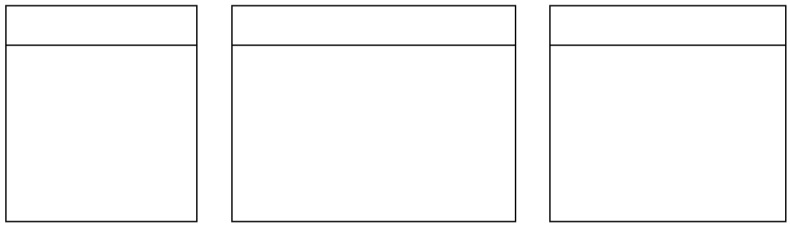
As cores são categorizadas por tonalidade (por exemplo, rosa, azul) e intensidade (por exemplo, pastel, quente). Algumas até mesmo brilham no escuro! O usuário pode querer examinar todos os produtos pastéis como um grupo, ou comparar produtos rosas *versus* azuis. A inclusão desses atributos na tabela de dimensão conforme mostra a Figura 8.7 pode acomodar essa necessidade.

As datas também formam hierarquias. Por exemplo, o usuário pode querer agrupar por mês, trimestre, ano ou dia da semana. As dimensões de data são muito comuns. O processo de estimativa possui duas dimensões de data: a data de estimativa e a data de vitória. Normalmente, as dimensões de data possuem atributos semelhantes. Há uma vantagem na padronização das dimensões de data pela empresa. Kimball e Ross [2002] recomendam estabelecer uma única dimensão de data padrão e depois criar visões da dimensão de data para uso em múltiplas dimensões. O uso de visões fornece a padroniza-



salto
o.i. pág. 160

Figura 8.7 Dimensão de cor mostrando atributos.



salto
o.i. pág. 161

Figura 8.8 Dimensões de data mostrando atributos.

ção, ao mesmo tempo em que permite que os atributos sejam nomeados com apelidos para uso intuitivo quando várias dimensões de data estão presentes. A Figura 8.8 ilustra esse conceito com uma dimensão de data e duas visões chamadas Estimate Date (Data de Estimativa) e Win Date (Data de Vitória).

Vamos prosseguir com o processo de agendamento. O agendamento usa as horas calculadas pelo processo de estimativa para planejar a carga de trabalho exigida em cada máquina. As datas de destino são atribuídas a cada etapa de manufatura. O ticket da tarefa passa para a produção após o término do processo de agendamento.

A XYZ Widget, Inc. possui um sistema de coleta automática de dados (ADC - Automatic Data Collection) no primeiro andar. Cada ticket de tarefa possui um código de barra do número da tarefa designada. Cada máquina possui uma folha com códigos de barra representando as diversas operações dessa máquina. Cada funcionário possui um crachá com um código de barras representando esse funcionário. Quando um funcionário inicia uma operação, o código de barras da tarefa é escaneado, assim como o código de barras da operação e o código de barras do funcionário. O computador obtém a hora atual do sistema como a hora inicial. Quando uma operação começa, a operação anterior desse funcionário é automaticamente finalizada (um funcionário é incapaz de fazer mais de uma operação ao mesmo tempo). Quando o trabalho na tarefa do produto termina nessa máquina, o funcionário marca a tarefa como completa por meio do sistema ADC. A informação colhida pelo sistema ADC é usada para atualizar o agendamento, rastrear as horas de trabalho e a produtividade do funcionário, e também rastrear a produtividade da máquina.

O projeto de um star schema do processo de agendamento começa determinando a granularidade. A tabela de agendamento mais detalhada no sistema operacional possui um registro para cada centro de custo que se aplica à manufatura de cada tarefa. Os usuários no departamento de agendamento

estão interessados em esmiuçar até esse nível de detalhe no data warehouse. O nível apropriado de granularidade no star schema para o agendamento é determinado pelo número da tarefa e o centro de custo.

Em seguida, determinamos as dimensões no star schema para o processo de agendamento. O sistema operacional de agendamento rastreia a data, hora de início e fim agendadas, assim como a data, hora de início e fim reais. As horas estimada e real também são armazenadas na tabela de detalhes de agendamento operacional, junto com uma marca indicando se a operação foi completada em tempo. A equipe de agendamento precisa ter a capacidade de agrupar registros pelas horas de início e fim agendadas e reais. Também é fundamental a capacidade de agrupar por centro de custo. As dimensões do star schema do agendamento são as datas e horas de início e fim agendada e real, e o centro de custo. O número da tarefa também precisa ser incluído como uma dimensão degenerada para manter a granularidade apropriada na tabela de fatos. A Figura 8.9 reflete as decisões sobre as dimensões apropriadas para o processo de agendamento.

A equipe de agendamento está interessada em agregar as horas estimadas e também as horas reais. Além disso, eles estão muito interessados em examinar tendências no desempenho de acordo com o tempo. As medidas apropriadas do agendamento no star schema incluem as horas estimada e real e uma marca indicando se a operação foi terminada em tempo. As medidas apropriadas do agendamento estão refletidas na Figura 8.9.

Existem vários princípios de padronização em jogo na Figura 8.9. Observe que existem várias dimensões de tempo., que devem ser padronizadas com

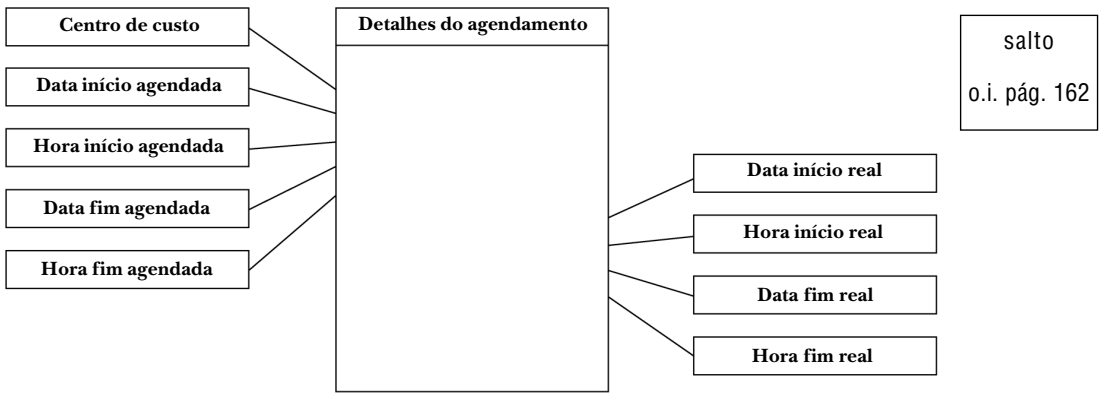


Figura 8.9 Star schema para o processo de agendamento.

uma única dimensão de tempo, junto com visões preenchendo os diferentes papéis, semelhante à técnica usada para as dimensões de data. Além disso, observe que a dimensão Centro de Custo está presente nos processos de estimativa e agendamento. Eles são, na realidade, os mesmos, e devem ser projetados como uma única dimensão. As dimensões podem ser compartilhadas entre vários star schemas. Um último ponto: as horas estimadas são transportadas da estimativa para o agendamento nos sistemas operacionais. Esses números são alimentados nos star schemas dos processos de estimativa e agendamento. O significado é o mesmo para os dois atributos; portanto, são chamados "horas estimadas". A regra prática é que, se dois atributos possuírem o mesmo significado, eles devem ter o mesmo nome, e se os dois atributos têm o mesmo nome, eles possuem o mesmo significado. Essa consistência permite a discussão e a comparação de informações entre os processos de negócios da empresa.

O próximo processo que examinamos é o rastreamento da produtividade. A granularidade é determinada pelo nível de detalhe disponível no sistema ADC. O detalhe inclui o número da tarefa, o centro de custo, o número do funcionário e a data e hora de início e fim. Os gerentes do departamento precisam ser capazes de agrupar linhas por centro de custo, funcionário e data e hora de início e fim. Esses atributos, portanto, se tornam as dimensões do star schema para o processo de produtividade, mostrado na Figura 8.10. Os gerentes estão interessados em agregar números de produtividade, incluindo a quantidade de produtos produzidos, a porcentagem terminada em tempo e as horas estimada e real. Como esses atributos devem ser agregados, eles se tornam as medidas mostradas na Figura 8.10.

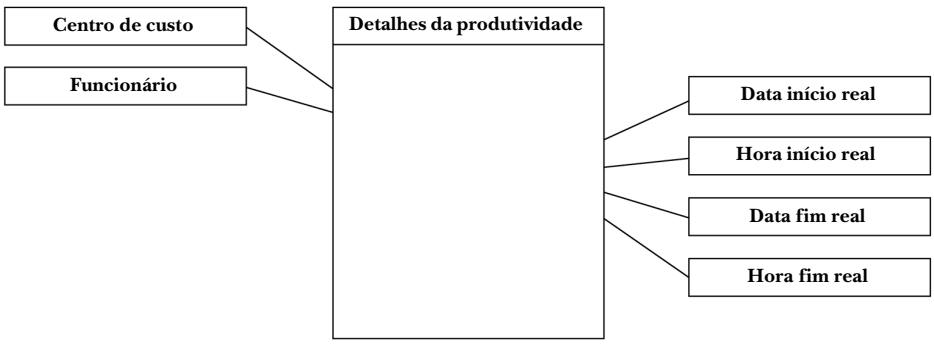


Figura 8.10 Star schema para o processo de rastreamento da produtividade.

Normalmente, existem dimensões em comum entre star schemas de um data warehouse, pois os processos de negócios normalmente estão interligados. Uma ferramenta útil para rastrear a semelhança e as diferenças de dimensões nos vários processos de negócios é o data warehouse bus [Kimball e Ross, 2002]. A Tabela 8.2 mostra um data warehouse bus dos quatro processos de negócios do nosso exemplo de projeto dimensional. Cada linha representa um processo de negócios. Cada coluna representa uma dimensão. Cada X no corpo da tabela representa o uso da dimensão indicada no processo de negócios indicado. O data warehouse bus é um meio prático de apresentar a organização de um data warehouse em um nível elevado. As dimensões comuns entre vários processos de negócios precisam ser padronizadas, ou "conformadas", na terminologia de Kimball e Ross. Uma dimensão é conformada se houver uma versão mais detalhada dessa dimensão e todos os outros usos dessa dimensão utilizarem um subconjunto dos atributos e um subconjunto das linhas da versão mais detalhada. A conformidade de dimensões garante que, sempre que os dados forem relacionados ou comparados pelos processos de negócios, o resultado será significativo.

O data warehouse bus também torna mais óbvias algumas decisões de projeto. Tomamos a liberdade de escolher as dimensões para o processo de custeio de tarefa. A Tabela 8.2 inclui uma linha desse processo. Quando você compara as linhas para estimar e custear a tarefa, rapidamente se torna claro que os dois processos têm a maioria das mesmas dimensões. Provavelmente, faz sentido combinar esses dois processos em um único star schema. Isso é especialmente verdadeiro, porque a análise de custeio de tarefa requer a comparação dos valores estimado e real. A Figura 8.11 apresenta o resultado da combinação dos processos de estimativa e custeio de tarefa em um único star schema.

Resumindo dados

Os star schemas que vimos até aqui são excelentes para capturar os detalhes pertinentes. Ter uma boa granularidade à disposição na tabela de fatos permite que os usuários examinem os dados até esse nível de granularidade. No entanto, os usuários normalmente desejam resumos. Por exemplo, os gerentes normalmente podem procurar um instantâneo diário dos dados de custeio de tarefa. Cada consulta que o usuário pode querer fazer num determinado star schema pode ser respondida a partir da tabela de fatos detalhada. O resumo poderia ser agregado no ato a partir da tabela de fatos. Existe uma desvantagem óbvia nessa estratégia. A tabela de fatos contém muitos

Tabela 8.2 Data warehouse bus para o exemplo de produto

	Forma	*Cor*	*Textura*	*Densidade*	*Tamanho*	*Data estimada*	*Data vitória*	*Cliente*	*Promoção*	*Centro de custo*	*Data início agendada*	*Hora início agendada*	*Data fim agendada*	*Hora fim agendada*	*Data início real*	*Hora início real*	*Data fim real*	*Hora fim real*	*Funcionário*	*Data da fatura*
Estimativa	X	X	X	X	X	X	X	X	X	X										
Agendamento										X	X	X	X	X	X	X	X			
Rastreamento de produtividade										X					X	X	X	X	X	
Custeio de tarefa	X	X	X	X	X			X	X	X										X

milhões de linhas, devido à natureza detalhada dos dados. A produção de um resumo na hora pode ser dispendiosa em termos de recursos de computador, resultando em uma resposta muito lenta. Se uma tabela de resumo estivesse disponível para responder as consultas para o instantâneo diário de custeio

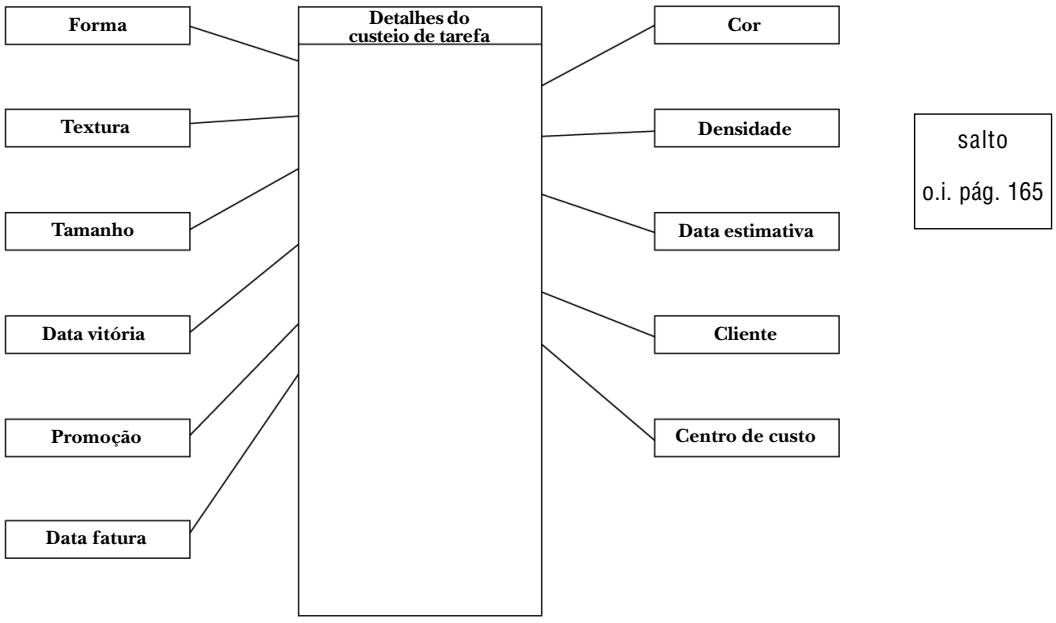


Figura 8.11 Diagrama star schema do processo de custeio de tarefa.

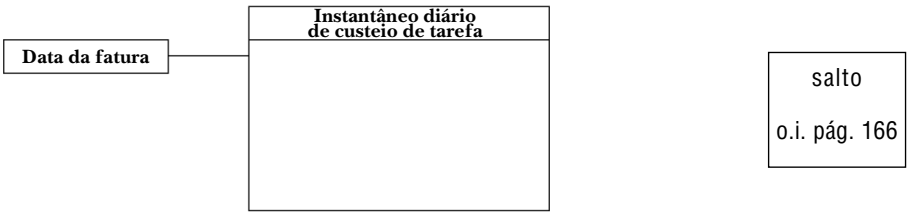


Figura 8.12 Esquema para o instantâneo diário de custeio de tarefa.

de tarefa, então a resposta poderia ser apresentada ao usuário de modo incrivelmente rápido. O esquema para o instantâneo diário de custeio de tarefa aparece na Figura 8.12. Observe que a maioria das dimensões usadas no detalhe de custeio de tarefa não é usada no instantâneo. O resumo dos dados eliminou a necessidade da maioria das dimensões nesse contexto. O instantâneo diário contém uma linha para cada dia em que as tarefas foram faturadas. O número de linhas no instantâneo estaria na casa de milhares. O pequeno tamanho do instantâneo permite uma resposta muito rápida quando um usuário solicita o instantâneo diário de custeio de tarefa. Quando existe um número pequeno de consultas resumos que ocorrem com freqüência, é uma boa estratégia materializar os dados resumos necessários para responder as consultas rapidamente.

O esquema de instantâneo diário na Figura 8.12 também permite que o usuário agrupe por mês, trimestre ou ano. Materializar dados de resumo é útil para a resposta rápida a qualquer consulta que possa ser vista agregando-se ainda mais os dados.

8.2 Online Analytical Processing (OLAP)

O projeto e a implementação de tabelas resumos estratégicas é uma boa técnica quando houver um pequeno conjunto de consultas freqüentes de dados resumos. Entretanto, pode haver a necessidade de alguns usuários explorarem os dados em um padrão ocasional. Por exemplo, um usuário que está procurando tipos de tarefas que não têm sido lucrativas precisa ser capaz de subir e descer várias dimensões dos dados. A natureza ocasional do processo torna impossível a previsão das consultas. O projeto de um conjunto estratégico de tabelas resumos para responder a essas explorações ocasionais dos dados é uma tarefa assustadora. OLAP oferece uma alternativa. OLAP é um serviço que sobrepõe o data warehouse. O sistema OLAP seleciona automati-

camente um conjunto estratégico de visões resumo e salva as tabelas resumo automaticamente (AST) em disco como visões normalizadas. O sistema OLAP também mantém essas visões, deixando-as alinhadas com as tabelas de fatos à medida que novos dados chegam. Quando um usuário solicita dados de resumo, o sistema OLAP descobre qual AST pode ser usada para dar resposta rápida à consulta indicada. Os sistemas OLAP são uma boa solução quando existe a necessidade de exploração ocasional das informações resumo com base em grandes quantidades de dados residindo no data warehouse.

Os sistemas OLAP automaticamente selecionam, mantêm e utilizam as ASTs. Assim, um sistema OLAP na verdade faz o mesmo trabalho de projeto automaticamente. Esta seção aborda alguns dos aspectos que surgem na montagem de um mecanismo OLAP e algumas das possíveis soluções. Se você usa um sistema OLAP, o vendedor entrega o mecanismo OLAP para você. Os problemas e soluções discutidos aqui não são itens que você precisa resolver. Nosso objetivo é acabar com parte do mistério sobre o que é um sistema OLAP e como ele funciona.

8.2.1 A explosão exponencial das visões

As visões materializadas agregadas a partir de uma tabela de fatos podem ser identificadas exclusivamente pelo nível de agregação para cada dimensão. Dada uma hierarquia ao longo de uma dimensão, considere que 0 representa nenhuma agregação, 1 representa o primeiro nível de agregação, e assim por diante. Por exemplo, se a dimensão Data da Fatura tiver uma hierarquia consistindo em id data, mês, trimestre, ano e "tudo" (ou seja, agregação completa), então id data é o nível 0, mês é o nível 1, trimestre é o nível 2, ano é o nível 3 e "tudo" é o nível 4. Se uma dimensão não tiver explicitamente uma hierarquia, então o nível 0 indicará que não possui nenhuma agregação, e o nível 1 será "tudo". As escalas assim definidas ao longo de cada dimensão definem um sistema de coordenadas para identificar exclusivamente cada visão em um gráfico de produtos. A Figura 8.13 ilustra um gráfico de produtos em duas dimensões. Os gráficos de produtos são generalizações da estrutura de hipercubo apresentada por Harinarayan, Rajaraman e Ullman [1996], na qual as dimensões podem ter hierarquias associadas. O nó superior, rotulado com (0, 0) na Figura 8.13, representa a tabela de fatos. Cada nó representa uma visão com níveis de agregação conforme indicado pela coordenada. Os relacionamentos de descendência do gráfico de produtos indicam relacionamentos de agregação. Os cinco nós sombreados indicam que essas visões foram

materializadas. Uma visão pode ser agregada a partir de qualquer visão ancestral materializada. Por exemplo, se um usuário emitir uma consulta por linhas agrupadas por ano e estado, essa consulta naturalmente seria respondida pela visão rotulada com (3, 2). A visão (3, 2) não é materializada, mas a consulta pode ser respondida a partir da visão materializada (2, 1), pois (2, 1) é ancestral de (3, 2). Os trimestres podem ser agregados em anos, e as cidades podem ser agregadas em estados.

O problema central que desafia o projeto de sistemas OLAP é a explosão exponencial de visões possíveis à medida que o número de dimensões aumenta. A dimensão Calendário na Figura 8.13 possui cinco níveis de hierarquia, e a dimensão Cliente tem quatro níveis de hierarquia. O usuário pode escolher qualquer nível de agregação ao longo de cada dimensão. O número de visões possíveis é o produto do número de níveis hierárquicos ao longo de cada dimensão. O número de visões possíveis do exemplo na Figura 8.13 é 5 x 4 = 20. Considere que *d* é o número de dimensões em um data warehouse. Consi-

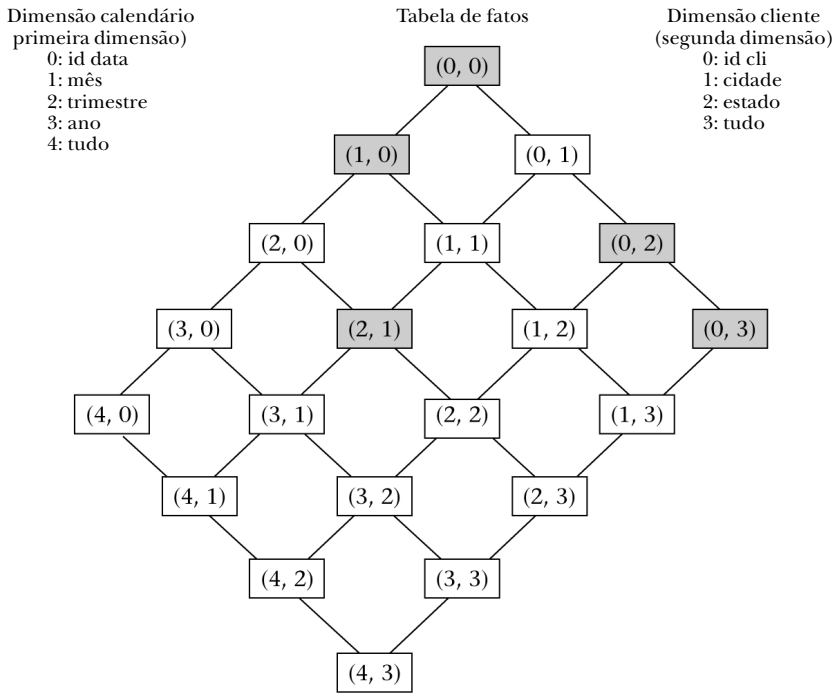


Figura 8.13 Gráfico de produtos rotulado com coordenadas de nível de agregação.

dere que h_i é o número de níveis hierárquicos na dimensão *i*. A equação geral para calcular o número de visões possíveis é dado pela Equação 8.1.

$$\text{Visões possíveis} = \prod_{i=1}^{d} h_i \qquad 8.1$$

Se expressarmos a Equação 8.1 em termos diferentes, o problema de explosão exponencial se tornará mais aparente. Considere que *g* é a média geométrica do número de níveis hierárquicos nas dimensões. Então a Equação 8.1 se torna a Equação 8.2.

$$\text{Visões possíveis} = g^d \qquad 8.2$$

À medida que a dimensionalidade aumenta linearmente, o número de visões possíveis explode exponencialmente. Se $g = 5$ e $d = 5$, existem $5^5 = 3.125$ visões possíveis. Assim, se $d = 10$, então existem $5^{10} = 9.765.625$ visões possíveis. Os administradores de OLAP precisam da liberdade de aumentar a dimensionalidade de seus data warehouses. Nitidamente, o sistema OLAP não pode criar e manter todas as visões possíveis à medida que a dimensionalidade aumenta. O projeto de sistemas OLAP precisa oferecer resposta rápida enquanto mantém o sistema dentro das limitações de recursos. Normalmente, um subconjunto estratégico das visões precisa ser selecionado para materialização.

8.2.2 Visão geral do OLAP

Existem muitas técnicas para implementar sistemas OLAP apresentadas na literatura. A Figura 8.14 mapeia uma técnica possível, que servirá de discussão. O maior problema da otimização OLAP é desmembrado em quatro subproblemas: estimativa do tamanho de visões, seleção de visões materializadas, manutenção de visões materializadas e otimização de consulta com visões materializadas. Essa divisão geralmente é aceita na literatura OLAP e se reflete no plano do sistema OLAP exibido na Figura 8.14.

Na Figura 8.14 descrevemos como os processos OLAP interagem a fim de explorar cada processo com mais detalhes. O plano de otimização do OLAP ilustra que os *Dados de amostra* movem-se da *Tabela de fatos* para a *Estimativa de tamanho da visão*. A *Seleção da visão* faz uma *Solicitação de estimativa*

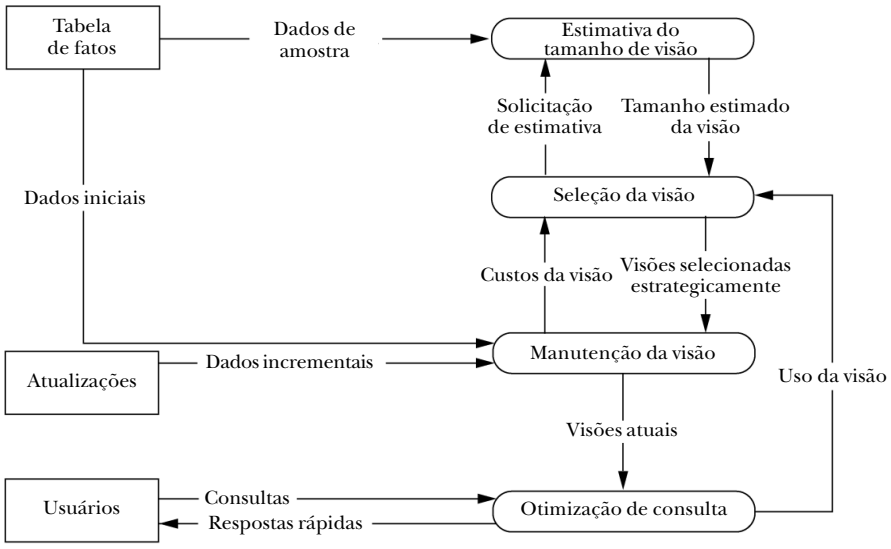


Figura 8.14 Um plano de otimização do OLAP.

do tamanho da visão para cada visão que ele considera para materialização. A *Estimativa de tamanho da visão* consulta os *Dados de amostra*, examina-os e modela a distribuição. A distribuição observada na amostra é usada para estimar o número esperado de linhas na visão para o conjunto de dados completo. A *Estimativa de tamanho da visão* é passada para *Seleção da visão*, que usa as estimativas para avaliar os benefícios relativos de materializar as diversas visões em consideração. A *Seleção da visão* utiliza as visões *selecionadas estrategicamente* para materialização com o objetivo de minimizar os custos totais da consulta. A *Manutenção da visão* monta as visões originais a partir dos *Dados iniciais* da *Tabela de fatos*, e mantém as visões à medida que os *Dados incrementais* chegam das *Atualizações*. *Manutenção da visão* retorna as estatísticas sobre *Custos da visão* apara a *Seleção da visão*, permitindo que as visões dispendiosas sejam descartadas dinamicamente. A *Manutenção das visões* fornece as *visões atuais* para que sejam usadas pela *Otimização de consulta*. A *Otimização de consulta* precisa considerar quais das *visões atuais* podem ser utilizadas para responder de modo mais eficiente as *Consultas* dos *Usuários*, dando *Respostas rápidas* aos *Usuários*. O *Uso da visão* alimenta a *Seleção da visão*, permitindo que o sistema se adapte dinamicamente às mudanças nas consultas de carregamento.

8.2.3 Estimativa de tamanho das visões

Os sistemas OLAP materializam seletivamente as visões estratégicas com os altos benefícios a fim de alcançar respostas rápidas às consultas, enquanto permanecem dentro dos limites de recursos do sistema computacional. O tamanho de uma visão afeta a quantidade de espaço em disco exigido para armazená-la. Mais importante, o tamanho da visão determina em parte a quantidade de entradas/saídas em disco consumida na consulta e manutenção da visão. O cálculo do tamanho exato de determinada visão é feito a partir dos dados básicos. A leitura dos dados básicos e o cálculo da visão é a maior parte do trabalho necessário para materializar a visão. Como o objetivo de materialização de visões é poupar recursos, torna-se necessário estimar o tamanho das visões consideradas para a materialização.

A fórmula de Cardenas [Cardenas, 1975] é uma equação simples (Equação 8.3) que se aplica à estimativa do número de linhas em uma visão:

Considere que n é o número de linhas na tabela de fatos.

Considere que v é o número de chaves possíveis no espaço de dados da visão.

$$\text{Valores distintos esperados} = v(1 - (1 - 1/v)^n) \qquad 8.3$$

A fórmula de Cardenas considera uma distribuição de dados uniforme. Contudo, existem muitas distribuições de dados. A distribuição de dados na tabela de fatos afeta o número de linhas em uma visão. A fórmula de Cardenas é muito rápida, mas a suposição de uma distribuição de dados uniforme leva a valores brutos superestimados do tamanho da visão quando os dados estão realmente agrupados. Outros métodos foram desenvolvidos para modelar o efeito da distribuição de dados sobre o número de linhas em uma visão.

Faloutsos, Matias e Silberschatz [1996] apresentam uma técnica de amostragem baseada na distribuição multifractal binomial. Os parâmetros da distribuição são estimados a partir de uma amostra. O número de linhas na visão agregada do conjunto completo de dados é estimado usando os valores de parâmetro determinados a partir da amostra. As Equações 8.4 e 8.5 [Faloutsos, Matias e Silberschatz, 1996] são apresentadas para essa finalidade.

$$\text{Valores distintos esperados} = \sum_{a=0}^{k} C_a^k (1 - (1 - P_a)^n) \qquad 8.4$$

$$P_a = P^{k-a}(1 - P)^a \qquad 8.5$$

A Figura 8.15 ilustra um exemplo. A ordem k é a profundidade da árvore de decisão. $C^k\ a$ é o número de componentes alcançáveis no conjunto tomando-se alguma combinação de a arestas esquerdas e k - a arestas direitas na árvore de decisão. *Pa* é a probabilidade de se alcançar determinado componente cujo caminho contém a bordas esquerdas. n é o número de linhas no conjunto de dados. Corte p é a probabilidade de selecionar a aresta direita em um ponto de escolha na árvore.

Os cálculos da Equação 8.4 são ilustrados com um pequeno exemplo. Um banco de dados real geraria números muito maiores, mas os conceitos e as equações são iguais. Esses cálculos podem ser feitos com logaritmos, resultando em uma escalabilidade muito boa. Com base na Figura 8.15, dadas cinco linhas, calcule os valores distintos esperados usando a Equação 8.4:

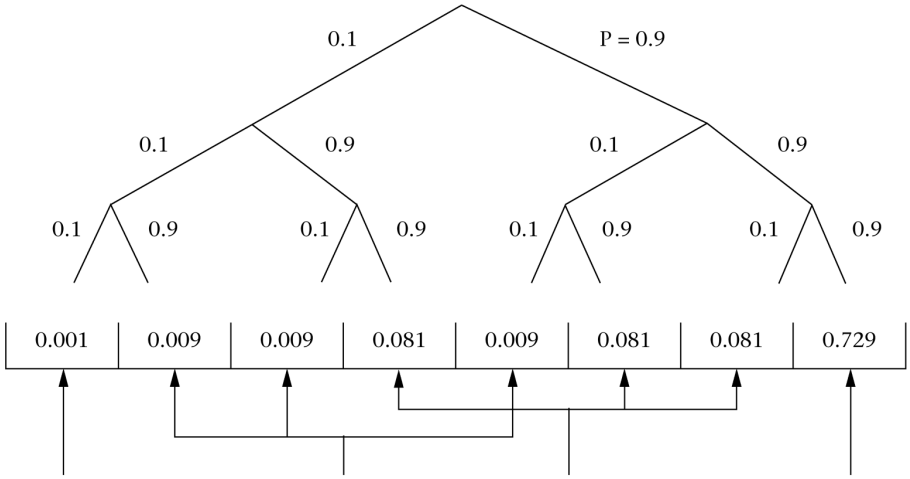
$$\begin{aligned}&\text{Valores distintos esperados} = \\ &1 \cdot (1 - (1 - 0.729)5) + 3 \cdot (1 - (1 - 0.081)5) + \\ &3 \cdot (1 - (1 - 0.009)5) + 1 \cdot (1 - (1 - 0.001)5)\ \text{H}1.965\end{aligned} \qquad 8.6$$

Os valores de P e k podem ser estimados com base nos dados de amostra. O algoritmo usado em [Faloutsos, Matias e Silberschatz, 1996] possui três entradas: o número de linhas no exemplo, a freqüência do valor que mais ocorre e o número de linhas agregadas distintas na amostra. O valor de P é calculado com base na freqüência do valor que ocorre com mais freqüência. Eles começam com:

$$k = [\text{Log}_2(\text{Linhas distintas na amostra})] \qquad 8.7$$

e depois ajustam k para cima, recalculando P até que seja encontrada uma boa escolha para o número de linhas distintas na amostra.

Outros modelos de distribuição podem ser utilizados para prever o tamanho de uma visão com base nos dados de amostra. Por exemplo, o uso do modelo de distribuição de Pareto tem sido explorado [Nadeau e Teorey, 2003]. Outra possibilidade é encontrar a melhor escolha dos dados de amostra para vários modelos de distribuição, calcular qual modelo tem maior probabilidade de produzir determinados dados de amostra e depois usar esse modelo para prever o número de linhas para o conjunto completo de dados. Isso



salto
o.i. pág. 172

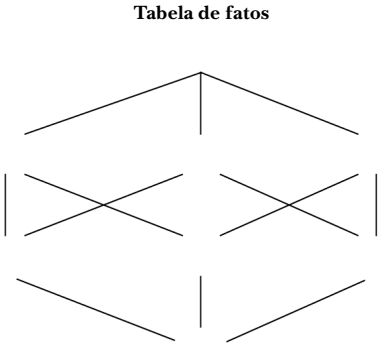
Figura 8.15 Exemplo de uma árvore de distribuição binomial multifractal.

exigiria o cálculo para cada modelo de distribuição considerado, mas geralmente resulta em estimativas mais precisas.

8.2.4 Seleção de visões materializadas

A maior parte dos trabalhos publicados sobre o problema de seleção de visões materializada é baseada na estrutura de hipercubo [Harinarayan, Rajaraman e Ullman, 1996]. A estrutura de hipercubo é um caso especial da estrutura de gráfico de produtos, em que o número de níveis hierárquicos para cada dimensão é dois. Cada dimensão pode ser incluída ou excluída de determinada visão. Assim, os nós em uma estrutura de hipercubo representam o conjunto de potência das dimensões.

A Figura 8.16 ilustra a estrutura de hipercubo com um exemplo [Harinarayan, Rajaraman e Ullman, 1996]. Cada nó da estrutura representa uma visão possível. Cada nó é rotulado com o conjunto de dimensões da lista "group by" dessa visão. Os números associados com os nós representam o número de linhas na visão. Esses números normalmente são derivados de um algoritmo de estimativa de tamanho de visão, conforme discutido na Seção 8.2.3. No



salto
o.i. pág. 173

Figura 8.16 Exemplo de uma estrutura de hipercubo [Harinarayan *et al.* 1996]

entanto, os números na Figura 8.16 seguem o exemplo dado por Harinarayan *et al.* [1996]. Os relacionamentos entre os nós indicam quais visões podem ser agregadas a partir de outras visões. Uma visão qualquer pode ser calculada a partir de qualquer visão ancestral materializada.

Chamamos o algoritmo de selecionar visões materializadas introduzido por Harinarayan *et al.* [1996] como HRU. O estado inicial para HRU tem apenas a tabela de fatos materializada. O HRU calcula o benefício de cada visão possível durante cada iteração e seleciona a visão mais benéfica para materialização. O processamento continua até que um número predeterminado de visões materializadas seja atingido.

A Tabela 8.3 mostra os cálculos para as duas primeiras iterações do HRU. A materialização {*p, s*} salva 6M – 0.8M = 5.2M linhas para cada uma das quatro visões: {*p, s*} e seus três descendentes: {p}, {s} e {}. A visão {*c, s*} não gera benefício materializado, pois qualquer consulta que puder ser respondida lendo-se 6M linhas de {*c, s*} também poderá ser respondida lendo-se 6M linhas da tabela de fatos {c, p, s}. O HRU calcula os benefícios de cada materialização de visão possível. A visão {*p, s*} é selecionada para materialização na primeira iteração. A visão {*c*} é selecionada na segunda iteração.

HRU é um algoritmo voraz que não garante uma solução ideal, embora o teste tenha mostrado que ele normalmente produz uma boa solução. Muitas pesquisas têm sido feitas sobre HRU levando-se em consideração a presença de estruturas de índice, custos de atualização e freqüências de consulta.

O HRU avalia cada nó não selecionado durante cada iteração, e cada avaliação considera o efeito sobre cada descendente. O algoritmo consome $O(kn^2)$ tempo, onde $k = |\text{views a selecionar}|$ e $n = |\text{nós}|$. Essa ordem de complexidade

Tabela 8.3 Duas iterações de HRU, com base na Figura 8.16

	Benefício da iteração 1	***Benefício da Iteração 2***
{p, s}	**5.2M x 4 = 20.8M**	
{c, s}	0 x 4 = 0	0 x 2 = 0
{c, p}	0 x 4 = 0	0 x 2 = 0
{s}	5.99M x 2 = 11.98M	0.79M x 2 = 1.58M
{p}	5.8M x 2 = 11.6M	0.6M x 2 = 1.2M
{c}	5.9M x 2 = 11.8M	**5.9M x 2 = 11.8M**
{}	6M – 1	0.8M – 1

parece ser muito boa; ela é polinomial no tempo. Todavia, o resultado é ilusório. Os nós da estrutura de hipercubo constituem um conjunto de potência. O número de visões possíveis é, portanto, 2^d, onde d = |dimensões|. Assim, $n = 2^d$, e a complexidade de tempo do HRU é $O(k2^{2d})$. O HRU é executado em um tempo exponencial em relação ao número de dimensões no banco de dados.

O Polynomial Greedy Algorithm (PGA) [Nadeau e Teorey, 2002] oferece uma alternativa mais viável que o HRU. PGA, assim como HRU, também seleciona uma visão para materialização em cada iteração. Contudo, o PGA divide cada iteração em uma fase de nomeação e uma fase de seleção. A primeira fase nomeia as visões promissoras em um conjunto de candidatas. A segunda fase estima os benefícios da materialização de cada candidata e seleciona a visão com a mais alta avaliação para materialização.

A fase de nomeação começa no topo da estrutura; na Figura 8.16, esse é o nó {*c*, *p*, *s*}. PGA nomeia o menor nó dentre os filhos. O conjunto candidato agora é {{*p*, *s*}}. O PGA então examina os filhos de {*p*, *s*} e nomeia o menor filho, {*s*}. O processo se repete até que o final da estrutura seja alcançado. O conjunto de candidatas é, então, {{*p*, *s*}, {*s*}, {}}. Quando um caminho de visões candidatas tiver sido nomeado, o algoritmo entra na fase de seleção. Os cálculos resultantes são mostrados nas Tabelas 8.4 e 8.5.

Compare as Tabelas 8.4 e 8.5 com a Tabela 8.3. Observe que o PGA faz menos cálculos do que o HRU e, neste exemplo, chega às mesmas decisões que o HRU. O PGA normalmente apanha um conjunto de visões quase tão benéficas quanto aquelas escolhidas pelo HRU, e o PGA ainda é capaz de funcionar quando o HRU falha devido à complexidade exponencial. PGA é

Tabela 8.4 Primeira iteração do PGA, baseada na Figura 8.16

Candidatos	*Benefício da iteração 1*
{p, s}	**5.2M X 4 = 20.8M**
{s}	5.99M X 2 = 11.98M
{}	6M – 1

polinomial em relação ao número de dimensões. Quando o HRU falha, o PGA estende a utilidade do sistema OLAP.

Os algoritmos de seleção de visão materializada discutidos até aqui são estáticos; ou seja, as visões são apanhadas uma vez e depois materializadas. Uma técnica totalmente diferente para a seleção de visões materializadas é tratar o problema semelhante ao gerenciamento de memória [Kotidis e Roussopoulos, 1999]. As visões materializadas constituem um pool de visões. Os metadados são rastreados no uso das visões. O sistema monitora tanto o espaço quanto as restrições da janela de atualização. O conteúdo do pool de visões é ajustado dinamicamente. Conforme as consultas são impostas, as visões são acrescentadas de maneira apropriada. Sempre que uma restrição é violada, o sistema seleciona uma visão para despejo. Assim, o pool de visões pode melhorar à medida que mais estatísticas de uso são reunidas. Esse é um sistema de auto-ajuste adequado aos padrões de consulta em mudança.

As técnicas estáticas e dinâmicas se complementam e devem ser integradas. As técnicas estáticas são executadas mais rapidamente a partir do início, mas não se adaptam. A seleção dinâmica de visões começa com um pool de visões vazio e, portanto, gera tempos de resposta lentos quando um data warehouse é carregado; porém, ela é adaptável e melhora com o tempo. A

Tabela 8.5 Segunda iteração do PGA, baseada na Figura 8.16

Candidatos	*Benefício da iteração 2*
{c, s}	0 X 2 = 0
{s}	0.79M X 2 = 1.58M
{c}	**5.9M X 2 = 11.8M**
{}	6M – 1

natureza complementar dessas duas técnicas influenciou nosso plano de projeto da Figura 8.14, conforme indicado por *Consultas* retroalimentando *Seleção da visão.*

8.2.5 Manutenção da visão

Quando uma visão é selecionada para materialização, ela precisa ser calculada e armazenada. Quando os dados básicos são atualizados, a visão agregada também precisa ser atualizada para manter a consistência entre as visões. A materialização da visão original e as atualizações incrementais são ambas consideradas como manutenção da visão na Figura 8.14. A eficiência da manutenção da visão é bastante afetada pelas estruturas de dados que implementam a visão. Os sistemas OLAP são multidimensionais, e as tabelas de fato contêm grandes quantidades de linhas. Os métodos de acesso que implementam o sistema OLAP precisam suplantar os desafios de alta dimensionalidade em combinação com grandes quantidades de linha. As estruturas físicas não serão tratadas aqui em detalhes.

A maior parte dos artigos de pesquisa na área de manutenção de visões considera que novos dados são periodicamente carregados com dados incrementais durante as janelas de atualização designadas. Normalmente, o sistema OLAP se torna indisponível aos usuários, enquanto os dados incrementais são carregados em massa, tirando proveito das eficiências das operações em massa. Há um lado ruim em adiar a carga de dados incrementais para a próxima janela de atualização. Se o data warehouse recebe dados incrementais uma vez por dia, então existe um período de latência de um dia.

Atualmente, existe uma tendência no setor para acomodar atualizações de dados quase em tempo real, mantendo o data warehouse alinhado com os sistemas operacionais. Isso às vezes é chamado de "data warehouse ativo" ou "analítico em tempo real". A necessidade de latência de dados de apenas alguns minutos apresenta novos problemas. Como estrutura de dados muito grandes podem ser mantidas de modo eficiente com pequenas entradas? Uma solução é ter um segundo conjunto de estruturas de dados com o mesmo esquema do data warehouse.

Esse segundo conjunto de estruturas de dados atua como um tanque de contenção para dados incrementais e é conhecido como cubo delta em terminologia OLAP. Os sistemas operacionais alimentam o cubo delta, que é pequeno e eficiente para mudanças incrementais rápidas. O cubo de dados é atualizado periodicamente a partir do cubo delta, tirando proveito das efi-

ciências da operação em massa. Quando o usuário consulta o sistema OLAP, a consulta pode ser emitida contra o cubo de dados e o cubo delta, para obter um resultado atualizado. O cubo delta fica escondido do usuário. O que o usuário vê é um sistema OLAP que é quase tão atualizado quanto os sistemas operacionais.

8.2.6 Otimização de consulta

Quando uma consulta é feita num sistema OLAP, pode haver múltiplas visões materializadas disponíveis que poderiam ser usadas para calcular o resultado. Por exemplo, se tivermos a situação representada na Figura 8.13, e um usuário emitir uma consulta para agrupar linhas por mês e estado, essa consulta será naturalmente respondida pela visão rotulada com (1, 2). Contudo, como (1, 2) não é materializada, precisamos encontrar um ancestral materializado para obter os dados. Existem três desses nós no gráfico de produtos da Figura 8.13. A consulta pode ser respondida a partir dos nós (0, 0), (1, 0) ou (0, 2). Com a possibilidade de responder a consultas de fontes alternativas, surge o problema de otimização quanto a qual fonte é a mais eficiente para determinada consulta. A maior parte das consultas existentes focaliza técnicas sintáticas. As traduções de consulta possíveis são executadas, os custos de consulta alternativa são estimados e o que parece ser o melhor plano é executado. Outra técnica é consultar uma tabela de metadados contendo informações sobre as visões materializadas para determinar a melhor visão para consultar, e depois traduzir a consulta SQL original para usar a melhor visão.

Os sistemas de banco de dados contêm tabelas de metadados que mantêm dados sobre as tabelas e outras estruturas usadas pelo sistema. As tabelas de

Tabela 8.6 Exemplo de metadados da visão materializada

Dimensões			
Calendário	***Cliente***	***Blocos***	***IDdaVisão***
	0	10.000.000	1
0	2	50.000	3
0	3	1.000	5
1	0	300.000	2
2	1	10.000	4

metadados facilitam o sistema em suas operações. Veja um exemplo no qual a tabela de metadados pode facilitar o processo de localizar a melhor visão para responder a uma consulta em um sistema OLAP. O sistema de coordenadas definido pelos níveis de agregação forma a base para organizar os metadados para rastrear as visões materializadas. A Tabela 8.6 apresenta os metadados para as visões materializadas sombreadas na Figura 8.13. As duas dimensões rotuladas com *Calendário* e *Cliente* formam a chave composta. A coluna *Blocos* rastreia o número real de blocos em cada visão materializada. A coluna *IDdaVisão* é usada para identificar a visão materializada associada. A implementação armazena visões materializadas como tabelas em que o valor do *IDdaVisão* faz parte do nome da tabela. Por exemplo, a linha com *IDdaVisão* = 3 contém informações sobre a visão agregada que é materializada como tabela **AST3** (abreviação de Automatic Summary Table 3).

Observe o padrão geral nas coordenadas das visões do gráfico de produtos em relação a relacionamentos ancestrais. Considere que Value(*V*, *d*) representa uma função que retorna o nível de agregação para a visão *V* junto com a dimensão *d*. Para duas visões quaisquer *Vi* e *Vj* onde *Vi* ? *Vj*, *Vi* é um ancestral de *Vj* se e somente se, para cada dimensão *d* da chave composta, Value(*Vi*, *d*) = Value(*Vj*, *d*). Esse padrão nas chaves pode ser utilizado para identificar os ancestrais de determinada visão consultando os metadados. A semântica do gráfico de produtos é capturada pelos metadados, permitindo que o sistema OLAP procure semanticamente a melhor visão ancestral materializada consultando a tabela de metadados. Depois que a melhor visão materializada for determinada, o sistema OLAP pode reescrever a consulta original para utilizar a melhor visão materializada e prosseguir.

8.3 Data mining

Duas técnicas gerais são usadas para extrair conhecimento de um banco de dados. Na primeira, um usuário pode ter uma hipótese para confirmar ou refutar. Esse tipo de análise é feito com consultas de banco de dados padrão e análise estatística. A segunda técnica para extrair conhecimento é fazer com que o computador procure correlações nos dados e apresente hipóteses promissoras para que o usuário leve em consideração. Os métodos incluídos aqui são técnicas de data mining desenvolvidas nos campos de Machine Learning e Knowledge Discovery.

Os algoritmos de data mining tentam solucionar diversos problemas comuns. Um problema geral é a categorização: dado um conjunto de casos com

valores conhecidos para alguns parâmetros, classificar os casos. Por exemplo, dadas observações de pacientes, sugerir um diagnóstico. Outro tipo de problema geral é o agrupamento: dado um conjunto de casos, encontrar agrupamentos naturais dos casos. O agrupamento é útil, por exemplo, na identificação de segmentos de mercado. As regras de associação, também conhecidas como análise de índices de preço ao consumidor, são um outro problema comum. As empresas às vezes querem saber que itens freqüentemente são comprados juntos. Esse conhecimento é útil, por exemplo, quando são tomadas decisões sobre como dispor os produtos em um supermercado. Existem muitos tipos de data mining disponíveis. Han e Kamber [2001] abordam o data mining no contexto de data warehouses e sistemas OLAP. Mitchell [1997] é um valioso recurso, escrito sob o ponto de vista de aprendizagem de máquinas. Witten e Frank [2000] fazem um estudo sobre data mining, junto com um freeware escrito em Java, disponível no Web site Weka [*http://www.cs.waikato.ac.nz/ml/weka*]. O Web site Weka é uma boa opção para aqueles que querem experimentar e modificar os algoritmos existentes. Os principais fornecedores de banco de dados também oferecem pacotes de data mining que funcionam com seus bancos de dados.

Devido ao grande escopo do data mining, focalizamos duas formas de data mining: mining preditivo e de texto.

8.3.1 Preditivo

A previsão é uma forma de data mining em que as tendências são modeladas conforme o tempo, usando dados conhecidos, e as tendências futuras são previstas com base no modelo. Existem muitos modelos de previsão diferentes, com variados níveis de sofisticação. Talvez o modelo mais simples seja o modelo de linha dos mínimos quadrados. A linha de melhor ajuste é calculada a partir de pontos de dados conhecidos, usando o método dos mínimos quadrados. A linha é projetada para o futuro a fim de determinar as previsões. A Figura 8.17 mostra uma linha de mínimos quadrados para um conjunto de dados real. Os pontos cruzados (com dentes) representam os dados conhecidos reais. Os pontos circulares representam a linha dos mínimos quadrados. Quando a linha dos mínimos quadrados se projeta além dos pontos conhecidos, essa região representa as previsões. Os intervalos associados com as previsões em nossas figuras representam um intervalo de previsão de 90%. Ou seja, dado um intervalo, existe uma probabilidade de 90% de que o valor real, quando conhecido, se encontre nesse intervalo.

A técnica da linha dos mínimos quadrados atribui pesos iguais a cada ponto de dados conhecido ao criar o modelo. A tendência para cima prevista na Figura 8.17 não fornece qualquer consideração especial para a recente virada para baixo.

A suavização exponencial é uma técnica que atribui maior peso à história recente do que a história distante. A dupla suavização exponencial modela dois componentes: nível e tendência (daí a "dupla" suavização exponencial). À medida que os valores conhecidos mudam em nível e tendência, o modelo se adapta. A Figura 8.18 mostra as previsões feitas usando a dupla suavização exponencial, com base no mesmo conjunto de dados usado para calcular a Figura 8.17. Observe que a previsão agora é mais ligada à história recente.

A tripla suavização exponencial modela três componentes: nível, tendência e sazonalidade. É mais sofisticado do que a dupla suavização exponencial e oferece melhores previsões quando os dados realmente exibem comportamento sazonal. A Figura 8.19 mostra as previsões feitas pela tripla suavização exponencial, com base nos mesmos dados usados para calcular as Figuras 8.17 e 8.18. Observe que os intervalos de previsão são mais estreitos do que nas Figuras 8.17 e 8.18. Esse é um sinal de que os dados variam de forma

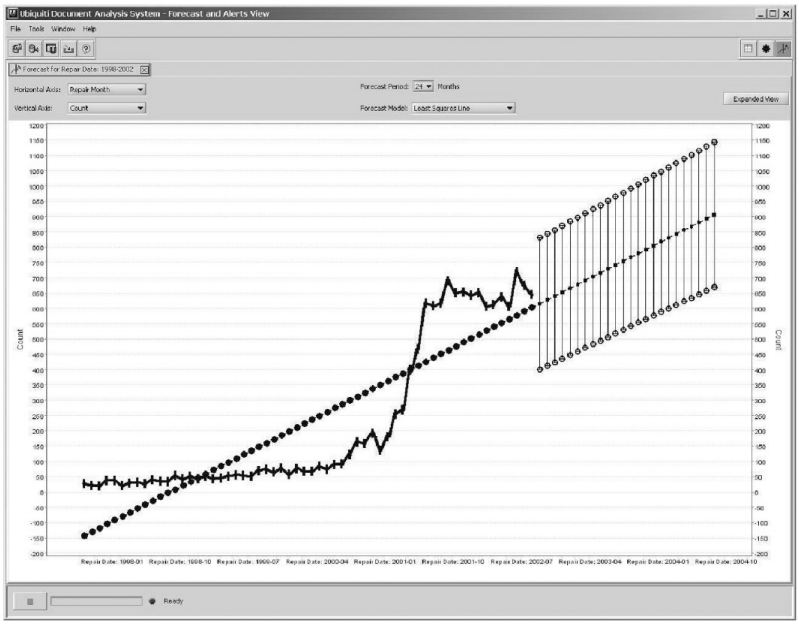


Figura 8.17 Linha dos mínimos quadrados (cortesia da Ubiquiti,Inc.).

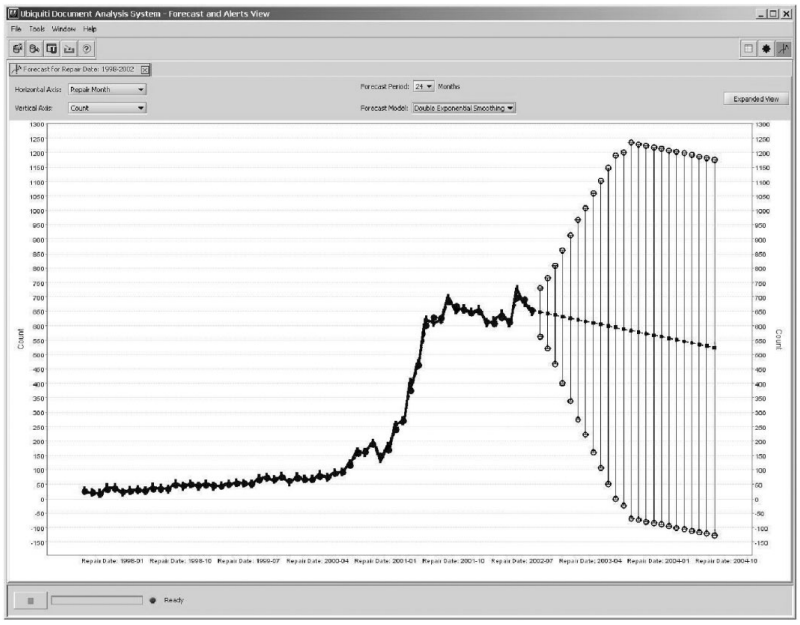


Figura 8.18 Dupla suavização exponencial (cortesia da Ubiquiti, Inc.).

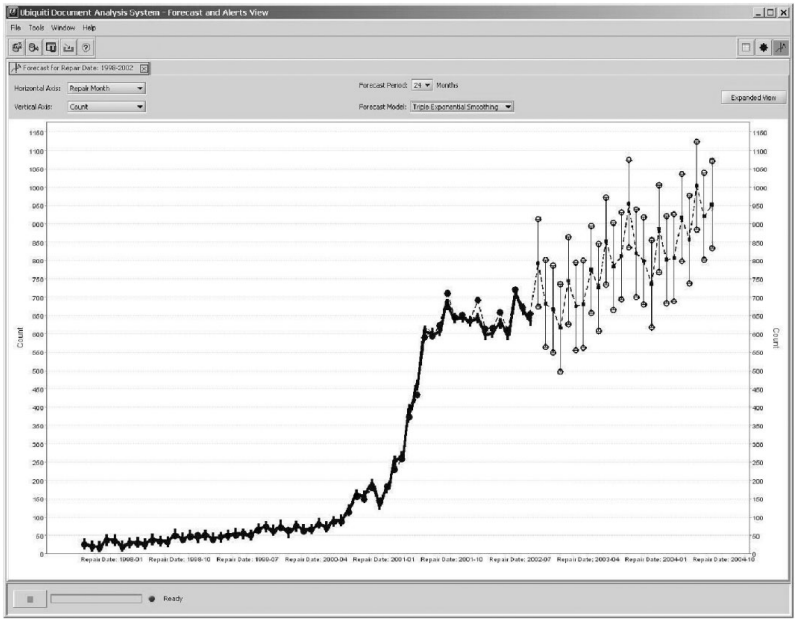


Figura 8.19 A suavização exponencial tripla (cortesia da Uniquiti, Inc.).

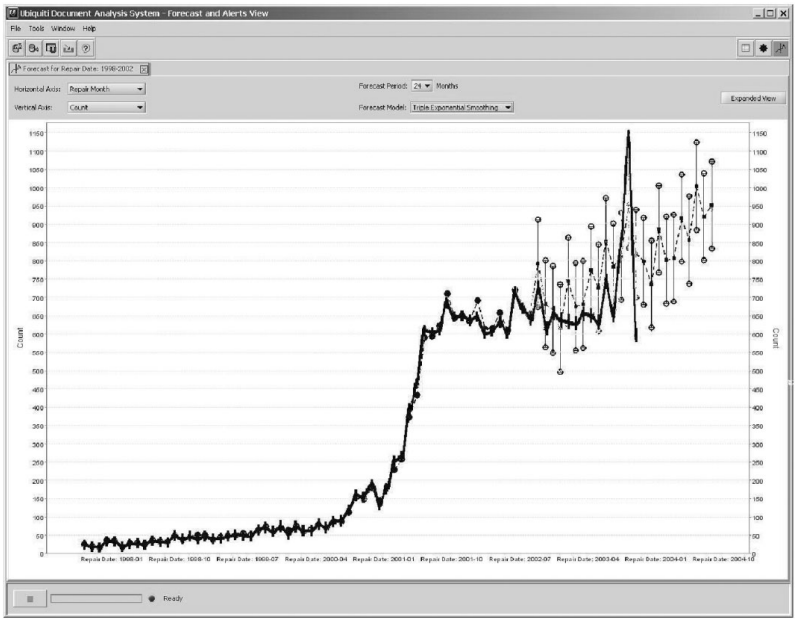


Figura 8.20 Tripla suavização exponencial com valores reais sobrepondo os valores de previsão, com base em cinco anos de dados de treinamento (cortesia da Ubiquiti, Inc.).

sazonal; a tripla suavização exponencial é um bom modelo para o tipo de dado indicado.

Quão confiáveis são essas previsões? Se retornarmos as previsões após o tempo ter se passado e compararmos as previsões com os valores reais, eles serão precisos? A Figura 8.20 mostra os dados reais sobrepostos com as previsões feitas na Figura 8.19. A maioria dos pontos de dados reais realmente se encontra dentro dos intervalos de previsão. Os intervalos de previsão parecem muito razoáveis. Por que não usamos esses modelos de previsão para ganhar milhões na Bolsa de Valores? Dê uma olhada na Figura 8.21, com cautela. A Figura 8.21 também é baseada no modelo de tripla suavização exponencial, usando quatro anos de dados conhecidos para treinamento, em comparação com cinco anos de dados usados na construção do modelo para a Figura 8.20. As previsões resultantes correspondem para quatro meses e depois divergem bastante da realidade. O problema é que os modelos de previsão são baseados em dados conhecidos, com a suposição de que os dados conhecidos formam uma boa base para prever o futuro. Isso pode ser verdadeiro na maior parte do tempo; porém, os modelos de previsão podem ser

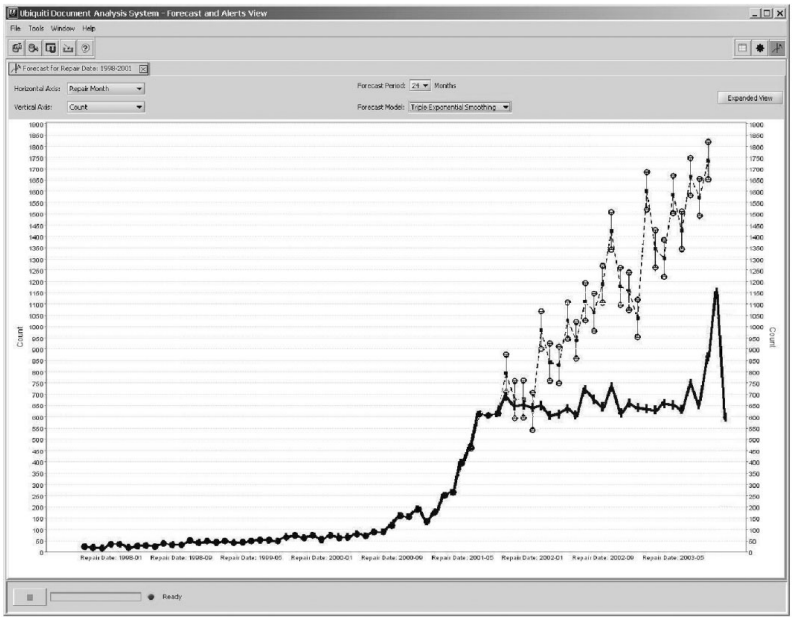


Figura 8.21 Tripla suavização exponencial com valores reais sobrepondo os valores de previsão, com base em quatro anos de dados de treinamento (cortesia da Ubiquiti, Inc.).

pouco confiáveis quando o mercado está mudando ou está para mudar drasticamente. A previsão pode ser uma ferramenta útil, mas as previsões precisam ser consideradas apenas como indicadores.

Os detalhes dos modelos de previsão discutidos aqui, além de muitos outros, podem ser encontrados em Makridakis *et al.* [1998].

8.3.2 Mining de Texto

A maior parte do trabalho em processamento de dados nas últimas décadas tem usado dados estruturados. A grande maioria dos sistemas em uso hoje lê e armazena dados em bancos de dados relacionais. Os esquemas são muito bem organizados em linhas e colunas. Entretanto, existem grandes quantidades de dados que residem em texto de forma livre. As descrições das reivindicações de garantia são escritas em texto. Registros médicos são escritos em texto. O texto está em toda a parte. Apenas recentemente o trabalho na análise de texto tomou rumo significativo. As empresas agora estão comercializando produtos que focalizam a análise de texto.

Vejamos algumas possibilidades para análise de texto e seu impacto em potencial. Pegaremos o setor de reivindicações de garantia automotiva como exemplo. Quando o carro tem algum problema, você deve levá-lo a uma oficina de automóveis para conserto. Você descreve a um representante da oficina o problema. Sua descrição é digitada em um computador. Um mecânico trabalha no carro e depois digita observações sobre ele e as ações tomadas para resolver o problema. Essa é uma informação valiosa para as empresas de automóveis e fabricantes de peças. Se as informações puderem ser analisadas, eles podem descobrir os problemas mais cedo e construir carros melhores. Podem reduzir as quebras, economizando dinheiro também e evitando frustração de seus clientes.

Os dados digitados no computador normalmente são inseridos com pressa. A linguagem inclui abreviações, jargão, palavras com erros de digitação e gramática incorreta. A Figura 8.22 mostra uma entrada de exemplo de um banco de dados real de reivindicação de garantia.

7 DD40 BASC 54566 CK OUT AC INOP PREFORM PID CK CK PCM PID ACC CK OK OPERATING ON AND OFF PREFORM POWER AND GRONED CK AT COMPRESOR FONED NO GRONED PREFORM PINPONT DIAG AND TRACE GRONED FONED BAD CO NECTION AT S778 REPAIR AND RETEST OK CK AC OPERATION

Figura 8.22 Exemplo de uma descrição literal em uma reivindicação de garantia (cortesia da Ubiquiti, Inc.).

Como você pode ver, a informação bruta inserida na oficina, mal pode ser considerada inglês. A Figura 8.23 mostra uma versão limpa do mesmo texto.

7 DD40 Basic 54566 Check Out Air Conditioning Inoperable Perform PID Check Check Power Control Module PID Accessory Check OK Operating On And Off Perform Power And Ground Check At Compressor Found No Ground Perform Pinpoint Diagnosis And Trace Ground Found Bad Connection At Splice 778 Repair And Retest OK Check Air Conditioning Operation.

Figura 8.23 Versão limpa da descrição no pedido de garantia (cortesia da Ubiquiti, Inc.).

Até mesmo a versão limpa é difícil de ler. As empresas que pagam pela garantia desejam que cada reivindicação seja categorizada de várias maneiras,

a fim de rastrear quais problemas estão ocorrendo. Uma opção é contratar muitas pessoas para ler as reivindicações e determinar como cada uma deve ser categorizada. A categorização das reivindicações de forma manual é um trabalho tedioso. Uma opção mais viável, desenvolvida nos últimos anos, é aplicar uma solução de software. A Figura 8.24 mostra algumas das informações que podem ser colhidas automaticamente a partir do texto na Figura 8.22.

Codificação automatizada	Confiança
Primary Group: Electrical	90 %
Subgroup: Climate Control	85 %
Part: Connector 1008	93 %
Problem: Bad Connection	72 %
Repair: Reconnect	75 %
Location: Engin. Cmprt.	90 %

Figura 8.24 Informações úteis extraídas da descrição literal no pedido de garantia (cortesia da Ubiquiti, Inc.).

O software processa o texto e determina os conceitos provavelmente representados no texto. Essa não é uma simples busca de texto. Sinônimos são mapeados para o mesmo conceito. Algumas palavras são mapeadas para conceitos diferentes, dependendo do contexto. O software utiliza uma ontologia que relaciona palavras e conceitos entre si. Após cada garantia ser categorizada de várias maneiras, é possível obter informações agregadas úteis, como mostra a Figura 8.25.

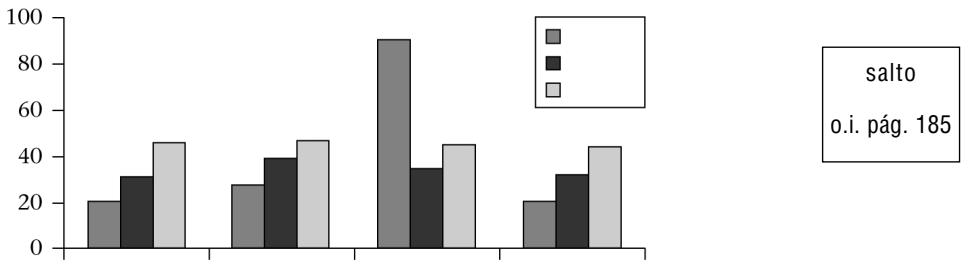


Figura 8.25 Dados agregados de pedidos de garantia (cortesia da Ubiquiti, Inc.).

8.4 Resumo

Data warehouse, OLAP e data mining são três áreas da ciência da computação que são altamente interligadas e comercializadas sob o título de business intelligence. As funcionalidades dessas três áreas se complementam entre si. O data warehouse oferece uma infra-estrutura para armazenar e acessar grandes quantidades de dados de uma maneira eficiente e amigável ao usuário. A modelagem de dados dimensional é a técnica mais adequada para projetar data warehouses. OLAP é um serviço que sobrepõe o data warehouse. A finalidade do OLAP é oferecer resposta rápida a consultas ocasionais, normalmente envolvendo o agrupamento de linhas e a agregação de valores. Operações de roll-up e drill-down são típicas. Sistemas OLAP realizam automaticamente algumas tarefas de projeto, como a seleção de quais visões materializar a fim de oferecer tempos de resposta rápidos. OLAP é uma boa ferramenta para explorar os dados em um padrão voltado para o ser humano, quando a pessoa possui uma pergunta clara em mente. Data mining normalmente é voltado para o computador, envolvendo análise dos dados para criar hipóteses prováveis que poderiam ser de interesse para os usuários. O data mining pode trazer grandes benefícios e uma estrutura de dados valiosa e interessante, que de outra forma passaria despercebida.

8.5 Resumo da literatura

A evolução e os princípios dos data warehouses podem ser encontrados em Barquin e Edelstein [1997], Cataldo [1997], Chaudhuri e Dayal [1997], Gray e Watson [1998], Kimball e Ross [1998, 2002] e Kimball e Caserta [2004]. OLAP é discutido em Barquin e Edelstein [1997], Faloutsos, Matia e Silberschatz [1996], Harinarayan, Rajaraman e Ullman [1996], Kotidis e Roussopoulos [1999], Nadeau e Teorey [2002 2003] e Thomsen [1997]. Os princípios e as ferramentas de data mining podem ser encontrados em Han e Kamber [2001], Makridakis, Wheelwright e Hyndman [1998], Mitchell [1997], The University of Waikato [2005], Witten e Frank [2000], entre muitos outros.

9 Ferramentas CASE para projeto lógico de banco de dados

O projeto de banco de dados é apenas uma parte da fase de análise e projeto para a criação de um software de aplicação de negócios eficiente (ver Figura 9.1), mas normalmente é a parte mais desafiadora e mais crítica para o desempenho. Nos capítulos anteriores, exploramos as maneias clássicas de criar projetos de banco de dados eficientes e eficazes, incluindo a modelagem ER e a transformação dos modelos ER em construtores por meio de regras de transformação. Também examinamos a normalização, formas normais e desnormalização, além das topologias específicas usadas no warehousing, como o star schema. Toda essa informação pode lhe causar tontura!

Este capítulo focaliza as ferramentas disponíveis comercialmente para simplificar esses processos de projeto. Essas ferramentas de engenharia de sistemas auxiliadas por computador, ou CASE (Computer-Aided System Engineering), oferecem funções que auxiliam no projeto do sistema. As ferramentas CASE são bastante usadas em diversos setores e domínios, como projeto de circuitos, manufatura e arquitetura. O projeto lógico de banco de dados é outra área na qual as ferramentas CASE têm provado sua eficácia. Este capítulo explora os principais fornecedores desse segmento: IBM, Computer Associates e Sybase. Cada uma dessas empresas fornece tecnologias poderosas, cheias de recursos, para o desenvolvimento de projetos lógicos de banco de dados e sua transição para bancos de dados físicos.

Embora seja impossível apresentar informações sobre produtos de software sem alguma subjetividade e comentários, sinceramente tentamos discutir as capacidades desses produtos com o mínimo de parcialidade ou crítica. Além disso, é impossível descrever os recursos desses produtos em grandes níveis de

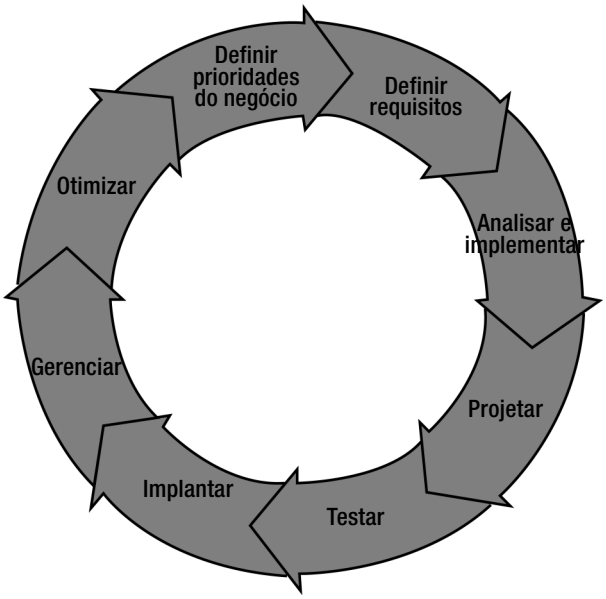


Figura 9.1 Ciclo de vida do sistema de negócios (cortesia da IBM Corp.).

detalhes num capítulo como este (um manual do usuário com centenas de páginas seria o mais adequado), de modo que estabelecemos um nível suficiente para dar ao leitor uma idéia das capacidades que oferecem. Outros detalhes podem ser obtidos nos Web sites dos fabricantes, que são apresentados no Resumo da Literatura, ao final deste capítulo.

9.1 Introdução às ferramentas CASE

Neste capítulo, apresentaremos alguns dos produtos mais populares e poderosos disponíveis no mercado para ajudar no projeto lógico do banco de dados: Rational Data Architect, da IBM, AllFusion ERwin Data Modeler, da Computer Associate, e PowerDesigner, da Sybase. Essas ferramentas CASE ajudam o projetista a desenvolver um banco de dados bem elaborado, acompanhando-o em um processo de projeto conceitual, projeto lógico e criação física, como mostra a Figura 9.2.

O AllFusion ERwin Data Modeler, da Computer Associates, existe há mais tempo. É um produto stand-alone, e seus pontos fortes vêm do apoio relativamente forte à modelagem física de dados e do maior conjunto de parceiros de tecnologia e treinamento terceirizados. O que ele faz, faz bem, mas nos

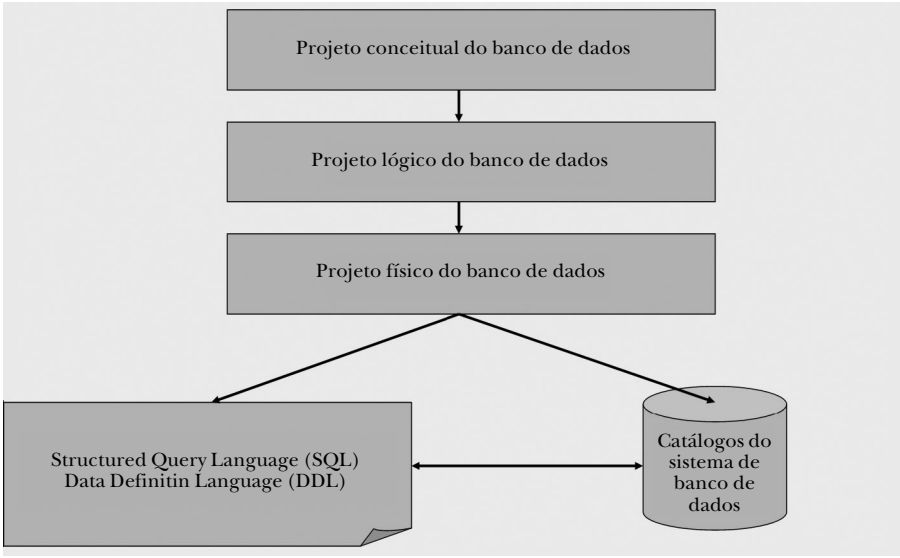


Figura 9.2 Processo de projeto de banco de dados.

últimos anos tem ficado para trás em alguns recursos avançados. O PowerDesigner, da Sybase, tem sido forte nos últimos anos, desafiando o AllFusion ERwin. Ele possui algumas vantagens nos relatórios e alguns recursos avançados, que serão descritos mais adiante neste capítulo. O Rational Data Architect, da IBM, é um produto novo, que suplanta o produto anterior da IBM, o Rational Rose Data Modeler. Seu ponto forte está na forte verificação de projeto; rica integração com a ampla plataforma de desenvolvimento de software da IBM, incluindo produtos de suas divisões Rational, Information Management e Tivoli; e recursos avançados que serão descritos mais adiante.

Nos capítulos anteriores, discutimos os aspectos do projeto lógico de banco de dados em que as ferramentas CASE podem ajudar a projetar, anotar, aplicar e modificar. Eles incluem, por exemplo, ER, modelagem UML e como essa modelagem pode ser usada no desenvolvimento de um projeto lógico de banco de dados. No projeto de relacionamentos da ER, existem vários tipos de definições de entidade e modelagem de relacionamentos (não relacionados, um-para-muitos e muitos-para-muitos). Esses relacionamentos são combinados e desnormalizados em padrões de esquemas conhecidos como formas normais (por exemplo, 3FN, star schema, esquema snowflake). Um projeto eficaz exige a definição clara de chaves, como as chaves primárias, estrangeiras e candidatas dentro dos relacionamentos. As restrições adicionais que limitam

o uso (e abusos) do sistema dentro de limites razoáveis ou regras de negócios também são fundamentais. O projeto lógico eficaz do banco de dados terá um impacto profundo sobre o desempenho do sistema, além da facilidade com que o sistema de banco de dados pode ser mantido e estendido.

Existem vários outros produtos CASE que não serão discutidos neste livro. Alguns produtos adicionais que merecem ser investigados são DeZign for Databases, da Datanamic, Qdesigner, da Quest Software, Visible Analyst by Standard e Embarcadero ER/Studio. O Visual Studio .NET Enterprise Architect edition inclui uma versão do Visio com alguns modelos de projeto de banco de dados que podem ser usados para criar modelos ER. O custo e a função dessas ferramentas variam bastante, desde produtos de fonte aberta até software empresarial completo, que custa milhares de dólares por licença.

O ciclo de desenvolvimento completo inclui um ciclo iterativo para conhecer os requisitos de negócios; definição dos requisitos do produto; análise e projeto; implementação; teste (componente, integração e sistema); implantação; administração e otimização; e gerenciamento de mudança. Nenhum produto único atual abrange esse escopo completamente. Em vez disso, os fornecedores de produtos oferecem, em graus variados, pacotes de produtos que se concentram em partes desse ciclo. Ferramentas CASE para o projeto de banco de dados focalizam bastante a análise e o projeto e, em menor grau, o teste do modelo e criação do banco de dados, conforme ilustrado na Figura 9.2.

Ferramentas CASE oferecem software que simplifica ou automatiza algumas das etapas descritas na Figura 9.2. O projeto conceitual inclui etapas como descrever as entidades de negócios e os requisitos funcionais do banco de dados; o projeto lógico inclui a definição de relacionamentos de entidades e formas normais; o projeto físico do banco de dados ajuda a transformar o projeto lógico em objetos de banco de dados reais, como tabelas, índices e restrições. As ferramentas de software fornecem valor significativo aos projetistas de banco de dados, pois:

1. Reduzem dramaticamente a complexidade do projeto conceitual e lógico; ambos podem ser muito difíceis de fazer. Essa redução na complexidade resulta em melhor projeto de banco de dados em menos tempo e com menor exigência de habilidade do usuário.
2. Automatiza a transformação do projeto lógico para o projeto físico (pelo menos o projeto físico básico). Isso reduz não apenas a exigência de tempo e habilidade do projetista, mas remove significativamente as chances de erro manual na conversão do modelo lógico para a linguagem de defi-

nição de dados física (DDL), que o servidor de banco de dados "consumirá" (como entrada) para criar o banco de dados físico.

3. Fornece relatórios, engenharia round trip e engenharia reversa, tornando essas ferramentas valiosas na manutenção de sistemas por um longo período. O projeto do sistema pode evoluir e evolui com o tempo, devido à alteração e expansão das necessidades de negócios. Além disso, as pessoas que projetam o sistema (às vezes, equipes de pessoas) podem não ser as mesmas encarregadas da manutenção do sistema. A complexidade de grandes sistemas, combinada com a necessidade de adaptabilidade contínua, praticamente exige o uso de ferramentas CASE para ajudar a visualizar, reverter a engenharia e rastrear o projeto do sistema com o tempo.

Você poderá encontrar uma lista maior de ferramentas de projeto de banco de dados disponíveis no Web site "Database Answers" (*http://www.databaseanswers.com/modelling_tools.htm*), mantido por David Alex Lamb na Queen's University em Kingston, Canadá.

9.2 Principais capacidades a observar

As ferramentas de projeto devem ser capazes de ajudá-lo na modelagem de dados e no projeto lógico do banco de dados. Os dois processos são importantes. Uma boa distinção entre estes processos aparece no Web site "Database Answers", citado anteriormente.

Na modelagem de dados, a pergunta a fazer é: com o que se parece o mundo que está sendo modelado? Em particular, você procura semelhanças entre coisas. Depois, você identifica um "supertipo" das coisas que são subtipos; por exemplo, Corporate Customers (Clientes Empresariais) e Personal Customers (Clientes — Pessoa Física). Se, por exemplo, os contatos do fornecedor forem coisas conceitualmente diferentes dos contatos do cliente, então a resposta é que eles devem ser modelados separadamente. Por outro lado, se forem simplesmente subconjuntos da mesma coisa, então devem ser tratados da mesma forma.

No projeto de banco de dados, você tenta responder a uma pergunta diferente: como posso projetar de forma eficiente um banco de dados que dará apoio às funções de uma aplicação ou Web site? A tarefa principal aqui é identificar as semelhanças entre as entidades, para que você possa integrá-las à mesma tabela, normalmente com um indicador "Type". Por exemplo, uma tabela Customer (Cliente), que combina todos os atributos de Corporate e Personal

Customers (Clientes Empresariais e Pessoa Física). Como resultado, é possível gastar muito tempo desmembrando as coisas quando se cria um modelo de dados, e depois juntá-las novamente ao projetar o banco de dados correspondente.

O apoio aos atributos de projeto programáveis e físicos com uma ferramenta de projeto também pode expandir o valor que uma ferramenta fornece. Em termos de banco de dados, os aspectos a observar incluirão representação de índices, controle de acesso, triggers e stored procedures.

As ferramentas de baixo nível (vendidas por menos de US$100 ou disponíveis como fonte aberta) fornecem as funcionalidades mais básicas da modelagem ER. Os produtos de mais alto nível fornecem os tipos de apoio necessários para um projeto sério, como:

- Engenharia round trip completa
- Projeto em UML
- Evolução do esquema; gerenciamento de mudança
- Engenharia reversa dos sistemas existentes
- Apoio à equipe, permitindo que várias pessoas trabalhem num mesmo projeto ao mesmo tempo
- Integração com Eclipse e .NET e outras ferramentas
- Reutilização de componentes e convenções (sendo capaz de reutilizar o padrão de nomeação, domínio e modelos lógicos em vários projetos)
- Recursos reutilizáveis (por exemplo, extensibilidade, modelos)
- Relatórios

9.3 Os fundamentos

Todos os produtos em questão fornecem funções robustas e de fácil utilização para a modelagem de dados e o projeto de banco de dados. Todos esses produtos fornecem a capacidade de representar graficamente os relacionamentos da ER. Essas ferramentas também fornecem os processos de transformação para mapear de um modelo ER para um projeto SQL (DDL), usando os tipos de transformação descritos anteriormente no Capítulo 5:

- Transformar cada entidade em uma tabela contendo os atributos chave e não-chave da entidade
- Transformar cada relacionamento binário muitos-para-muitos, recursivo ou não, em uma tabela com as chaves das entidades e os atributos do relacionamento

- Transformar cada relacionamento ternário ou *n*-ário de nível superior em uma tabela

De forma similar, essas ferramentas produzem os tipos de tabela de transformação descritos no Capítulo 5:

- Uma tabela entidade com o mesmo conteúdo de informação da entidade original
- Uma tabela entidade com a chave estrangeira embutida da entidade pai
- Uma tabela de relacionamentos com as chaves estrangeiras de todas as entidades envolvidas no relacionamento

O Capítulo 5 também descreveu regras para transformações de valores nulos que precisam ser aplicados, e as ferramentas CASE normalmente as impõem.

Essas ferramentas CASE também ajudam com a modelagem das formas normais e a desnormalização para desenvolver um esquema físico real para o seu banco de dados, conforme descrito no Capítulo 5. As ferramentas fornecem interfaces gráficas para o projeto físico de banco de dados, além da modelagem básica de exclusividade, restrições e índices. A Figura 9.3 mostra um exemplo da GUI (Graphical User Interface) do IBM Rational Data Architect para modelagem de ERs. A Figura 9.4 mostra um instantâneo semelhante da interface para o AllFusion ERwin Data Modeler, da Computer Associate.

Depois de criar um modelo ER, as ferramentas CASE permitem a sua fácil modificação por meio de interfaces gráficas. Um exemplo é apresentado a seguir na Figura 9.5, com o Rational Data Architect, da IBM, ilustrando a edição de atributos. Dessas ferramentas CASE, o Rational Data Architect talvez tenha a função de modelagem UML mais útil para modelagem e projeto de dados. Contudo, a notação IE-ERD é mais antiga e mais utilizada. Um dos aspectos interessantes do Rational Data Architect é a capacidade de trabalhar com a notação UML ou IE.

A Figura 9.6 mostra a tela do AllFusion ERwin para definir a cardinalidade dos relacionamentos de entidades. Vale a pena observar que muitos relacionamentos nem sequer precisam entrar nesse diálogo.

9.4 Gerando um banco de dados a partir de um projeto

Para realmente levar seu projeto ao próximo nível (ou seja, um nível prático), você precisará de uma boa ferramenta de projeto que possa interagir

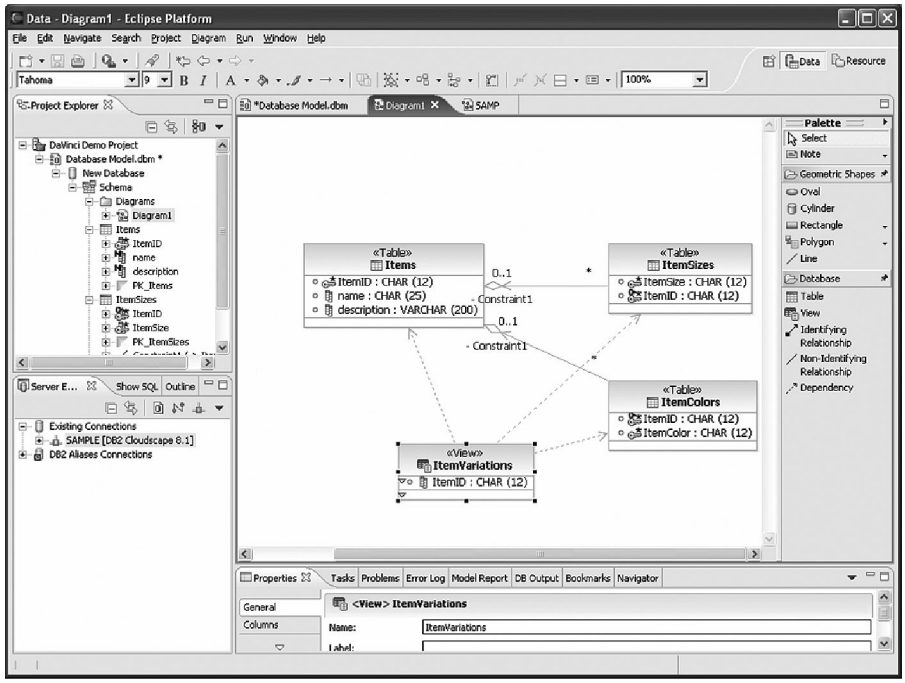


Figura 9.3 Modelagem ER do Rational Data Architect
(cortesia da IBM Rational Division)

com seu produto de banco de dados específico, de forma a realmente criar a Data Definition Language (DDL), e de scripts ou comandos associados, a fim de criar e modificar o banco de dados básico para você. Por exemplo, usando o exemplo do Capítulo 7, você modelou um modelo ER contendo os relacionamentos de vendas mostrados na Tabela 9.1.

Tabela 9.1 Modelo ER contendo relacionamentos de vendas

Construção ER	***DFs***
Customer(many): Job(one)	cust-no -> job-title
Order(many): Customer(one)	order-no -> cust-no
Salesperson(many): Department(one)	sales-name -> dept-no
Item(many): Department(one)	item-no -> dept-no
Order(many): Item(many): Salesperson(one)	order-no,item-no->sales-name
Order(many): Department(many): Salesperson(one)	order-no,dept-no-> sales-name

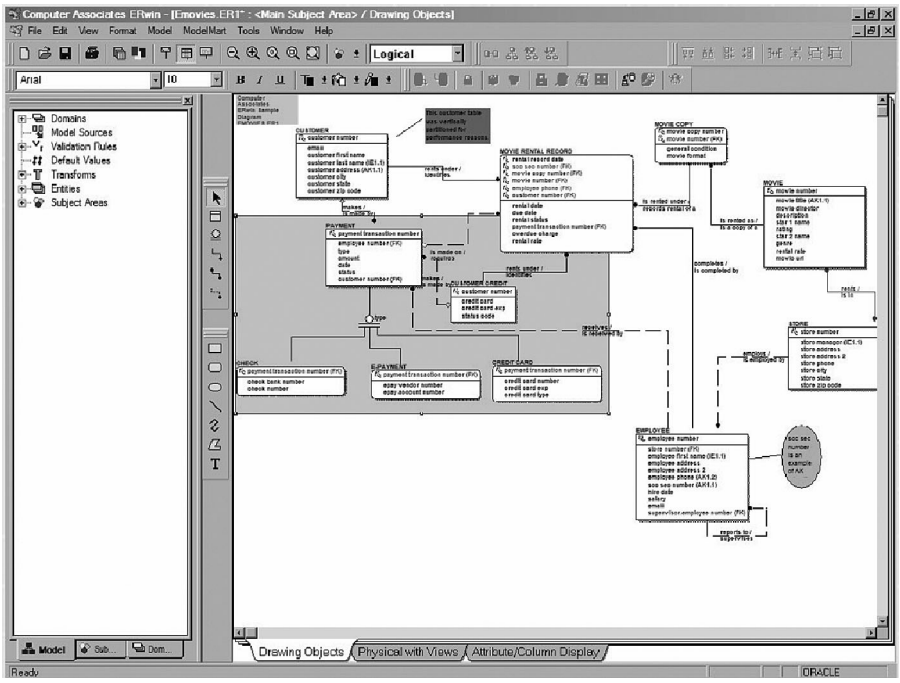


Figura 9.4 Modelagem de ER do AllFusion ERwin Data Modeler (figura da Computer Associates, *http://agendas.ca.com/Agendas/Presentations/AFM54PN.pdf*)

As ferramentas CASE automaticamente gerarão os scripts exigidos, incluindo a especificação DDL para criar o banco de dados real, e lhe darão uma opção para aplicar essas mudanças a um banco de dados real, da seguinte forma:

```
create table customer (cust_no char(6),
    job_title varchar(256),
    primary key (cust_no),
    foreign key (job_title) references job
    on delete set null on update cascade);

create table job (job_title varchar(256),
    primary key (job_title));
```

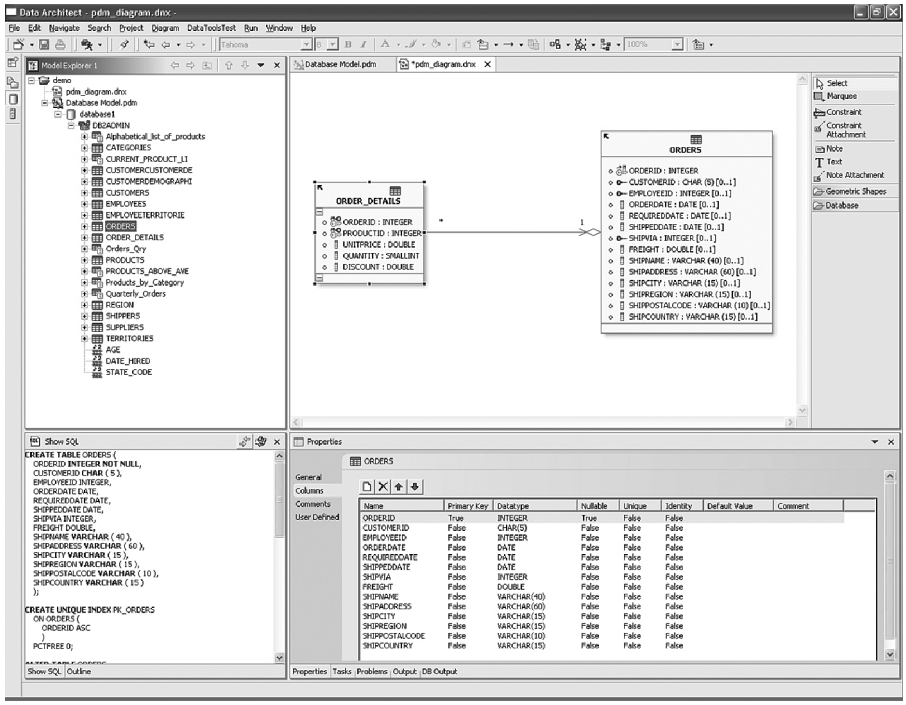


Figura 9.5 Edição de propriedades com o IBM Rational Data Architect (cortesia da IBM Rational Division)

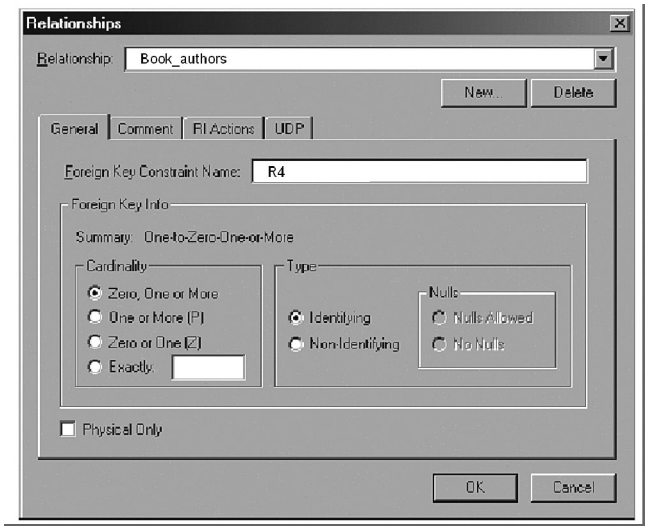


Figura 9.6 Especificando relacionamentos um-para-muitos com o ERwin (cortesia da Computer Associates)

```
create table order (order_no char(9),
    cust_no char(6) not null,
    primary key (order_no),
    foreign key (cust_no) references customer
    on delete cascade on update cascade);

create table salesperson (sales_name varchar(256),
    dept_no char(2),
    primary key (sales_name),
    foreign key (dept_no) references department
    on delete set null on update cascade);

create table department (dept_no char(2),
    primary key (dept_no));

create table item (item_no char(6),
    dept_no char(2),
    primary key (item_no),
    foreign key (dept_no) references department
    on delete set null on update cascade);

create table order_item_sales (order_no char(9),
    item_no char(6),
    sales_name varchar(256) not null,
    primary key (order_no, item_no),
    foreign key (order_no) references order
    on delete cascade on update cascade,
    foreign key (item_no) references item
    on delete cascade on update cascade,
    foreign key (sales_name) references salesperson
    on delete cascade on update cascade);

create table order_dept_sales (order_no char(9),
    dept_no char(2),
    sales_name varchar(256) not null,
    primary key (order_no, dept_no),
    foreign key (order_no) references order
    on delete cascade on update cascade,
```

```
foreign key (dept_no) references department
on delete cascade on update cascade,
foreign key (sales_name) references salesperson
on delete cascade on update cascade);
```

Vale a pena observar que, com todas as ferramentas CASE que discutimos aqui, a conversão do projeto lógico para o projeto físico é bastante rudimentar. Essas ferramentas ajudam a criar objetos de banco de dados básicos, como tabelas e, em alguns casos, índices. No entanto, os recursos avançados do servidor de banco de dados normalmente não são admitidos – e quando o são, a ferramenta CASE normalmente fica atrás por duas ou três versões de software. O desenvolvimento de recursos de projeto físico avançados, como agrupamento multidimensional ou visões materializadas, está muito além das capacidades das ferramentas de projeto lógico que estamos discutindo. O projeto avançado de banco de dados físico normalmente depende bastante da densidade dos dados e dos padrões de acesso aos dados. Um recurso do Rational Data Architect que se destaca é que ele oferece ligações com as capacidades de computação automáticas (autogerenciáveis) dentro do DB2 para oferecer seleção semi-automatizada dos atributos avançados de projeto físico.

A Figura 9.7 mostra um exemplo com a geração de esquema do ERwin, gerando a DDL DB2 diretamente a partir do modelo ER projetado dentro do ERwin.

Outras habilidades muito importantes compartilhadas por essas ferramentas incluem a capacidade de reverter a engenharia dos bancos de dados existentes (para os quais você pode não ter um modelo ER ou UML físico) e a capacidade de materializar automaticamente o modelo físico ou as mudanças incrementais de um modelo para um banco de dados real. Esta permite que você sincronize seu projeto de banco de dados com um banco de dados real enquanto faz mudanças. Essa capacidade é bastante útil para o desenvolvimento incremental de aplicação e banco de dados, além da manutenção incremental.

9.5 Apoio ao banco de dados

Todos esses produtos apóiam um grande conjunto de tipos de banco de dados. Certamente, todos os principais fornecedores de banco de dados são admitidos por cada um desses produtos (ou seja, DB2 UDB, DB2 zOS, Informix IDS, Oracle, SQL Server), e um conjunto muito maior é admitido por meio

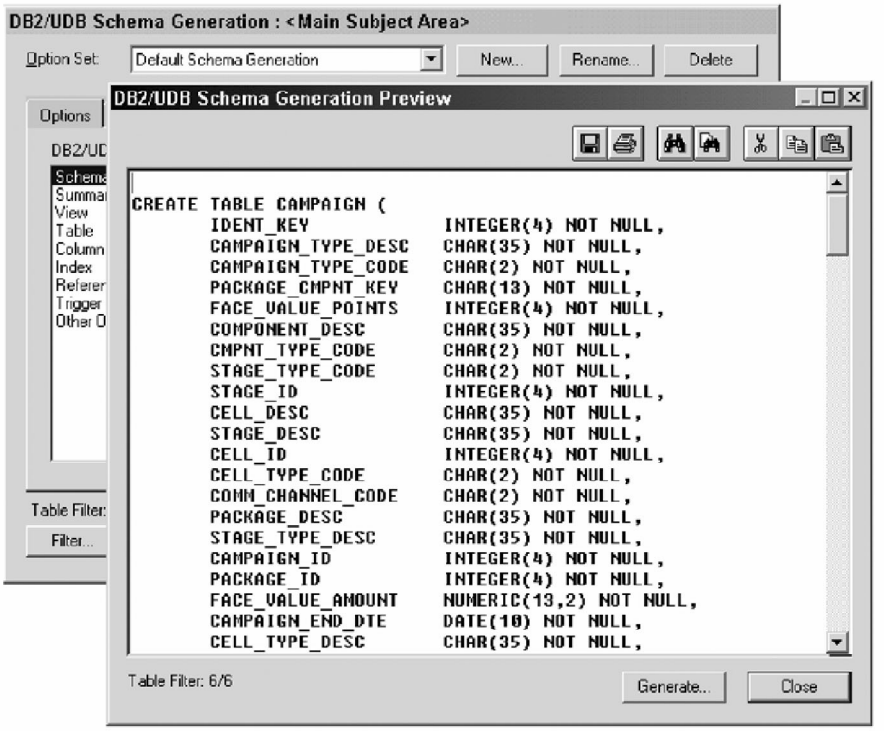


Figura 9.7 Geração de esquema do ERwin para um banco de dados DB2 (figura da IBM: *http://www.redbooks.ibm.com/abstracts/redp3714.html?Open*).

do ODBC. Entretanto, o que realmente importa mais para o desenvolvedor de aplicação é se o banco de dados programado é diretamente apoiado pela ferramenta CASE de projeto. O apoio ao banco de dados não é igual entre esses produtos. Além disso, de modo muito significativo, cada produto de banco de dados terá recursos próprios exclusivos (como índices de varredura reversa, restrições informativas e assim por diante), que não são padronizados. Um dos atributos qualitativos de uma boa ferramenta de projeto de banco de dados é se ela distingue e apóia as extensões exclusivas dos produtos de banco de dados individuais. Cada um dos produtos possui um forte histórico para fazer isso: em ordem decrescente, AllFusion ERwin Data Modeler, Sybase PowerDesigner e Rational Data Architect. Nitidamente, o Rational Data Architect da IBM tem uma abrangência um pouco menor de bancos de dados do que os outros produtos, embora admita todas as principais plataformas de banco de dados. Todavia, pode-se esperar que o Rational Data Architect, com o tempo, tenha a melhor integração com as famílias DB2 e Informix, pois

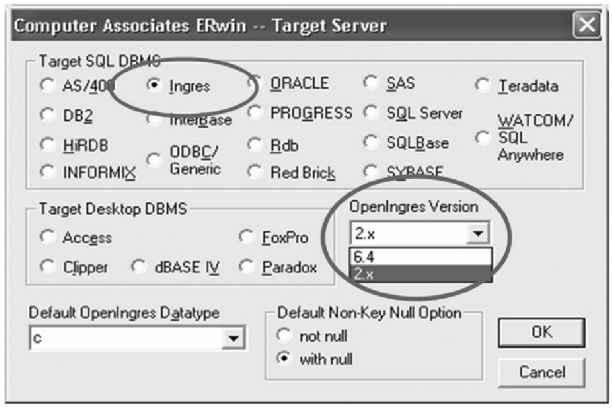


Figura 9.8 Seleção de DBMS no AllFusion ERwin Data Modeler (figura da Computer Associates: *http://iua.org.uk/conference/Autumn%202003/Ruth%20Wunderle.ppt#9*).

todos esses produtos são desenvolvidos pela IBM. Os projetistas de banco de dados são aconselhados a investigar o nível de apoio fornecido por uma ferramenta CASE para o banco de dados sendo desenvolvido, de modo a garantir o nível de apoio adequado. A Figura 9.8 mostra um exemplo de seleção de servidor de banco de dados com o AllFusion ERwin Data Modeler.

9.6 Apoio colaborativo

Todos esses três produtos são projetados para desenvolvimento colaborativo, de modo que vários desenvolvedores possam trabalhar juntos para projetar partes de um projeto de banco de dados, seja dando apoio a diferentes aplicações ou colaborando nas mesmas partes. Esses recursos de colaboração se encontram em dois domínios:

1. **Controle de concorrência.** Essa forma de colaboração garante que vários projetistas não modifiquem o mesmo componente do projeto de banco de dados ao mesmo tempo. Isso é comparável em termos de desenvolvimento de software a um sistema de controle de código fonte.
2. **Capacidades de integração e colaboração.** Essa forma de colaboração permite que os projetistas combinem projetos ou integrem suas últimas mudanças em um projeto maior. Essas capacidades de integração comparam componentes entre o que está sendo registrado no projeto do sistema e o

que um projetista deseja incluir ou modificar. As ferramentas CASE localizam as mudanças em conflito e as identificam visualmente para o projetista, o qual pode decidir quais mudanças devem ser mantidas e quais devem ser descartadas em favor do modelo atualmente definido no projeto.

A Figura 9.9 mostra a GUI de integração do Sybase PowerDesigner, que identifica mudanças significativas entre o esquema existente e o novo esquema sendo integrado. Em particular, observe como a ferramenta de integração identificou uma mudança na **Table_1** Column_1, que mudou os tipos básicos. A ferramenta também descobriu que **Table_2** e **Table_3**, que existem no projeto de integração, não estavam presentes no projeto básico. AllFusion ERwin Data Modeler e Rational Data Architect possuem capacidades semelhantes para integração de mudanças do projeto.

9.7 Desenvolvimento distribuído

O desenvolvimento distribuído tornou-se um fato comum para as equipes de desenvolvimento das grandes empresas, em que os grupos de desenvolvedores colaboram de locais geograficamente diversos para desenvolver um projeto. O fenômeno não é apenas verdadeiro entre os andares de um prédio, ou entre os diferentes locais em uma cidade, mas agora entre estados, províncias e até mesmo países. Na verdade, a terceirização do desen-

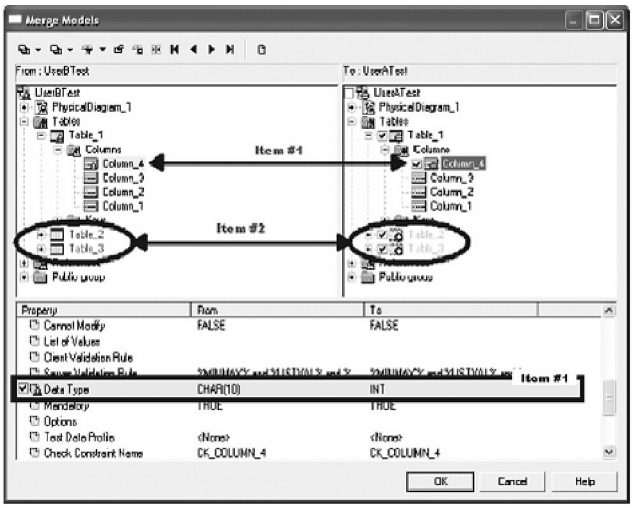


Figura 9.9 Processo de integração com o PowerDesigner (cortesia da Sybase).

volvimento de software tornou-se uma viagem sem volta;, muitos analistas prevêem que as empresas, em média, no futuro, terceirizarão 60% do trabalho de aplicação, deslocando os aspectos de desenvolvimento de projeto para locais com mão-de-obra mais barata. Conforme o META Group descreveu em seu relatório Offshore Market Milieu de 16 de setembro de 2004, "Com recursos globais custando um terço a um quinto do custo dos funcionários americanos - sem levar em conta os custos ocultos e maior disciplina de processo —, estratégias offshore agora permeiam as organizações de TI norte-americanas ".

Portanto, os desenvolvedores de software de banco de dados que trabalham em um ambiente colaborativo distribuído precisam considerar as qualidades de colaboração e distribuíção das ferramentas CASE para projeto de banco de dados. A tendência em direção ao desenvolvimento colaborativo e desenvolvimento distribuído não mostra sinais de atraso; em vez disso, está aumentando. No espaço existente, o software Rational MultiSite da IBM, mostrado na Figura 9.10, permite a melhor administração por locais geograficamente diversificados para replicar o software de projeto e dados e subseqüentemente integrar os resultados. O Rational MultiSite é uma tecnologia construída com base no Rational ClearCase e Rational ClearQuest (produtos de desenvolvimento e controle de código-fonte) para permitir réplicas locais dos repositórios do Rational ClearCase e Rational ClearQuest. O Rational MultiSite também lida com o sincronismo automático das réplicas. Isso é particularmente útil para empresas com desenvolvimento distribuído, que querem ter tempos de resposta rápidos para seus desenvolvedores por meio do acesso a informações locais, e essa replicação normalmente é um requisito absoluto para equipes distribuídas geograficamente.

9.8 Integração de ferramentas no ciclo de vida da aplicação

As melhores ferramentas CASE para projeto de banco de dados são integradas com um pacote completo de ferramentas de desenvolvimento de aplicação que abordam o ciclo de vida de desenvolvimento de software. Isso permite que toda a equipe de desenvolvimento trabalhe numa plataforma de ferramentas integradas, em vez de os modeladores de dados ficarem isolados em seu próprio mundo. Somente os maiores fornecedores oferecem isso, e, de fato, a verdadeira integração de ferramentas de acordo com o ciclo de vida do desenvolvimento é um tanto rara.

Essa solução é verdadeiramente a pedra filosofal dos fornecedores de infraestrutura de desenvolvimento. Todos os fornecedores que produzem plata-

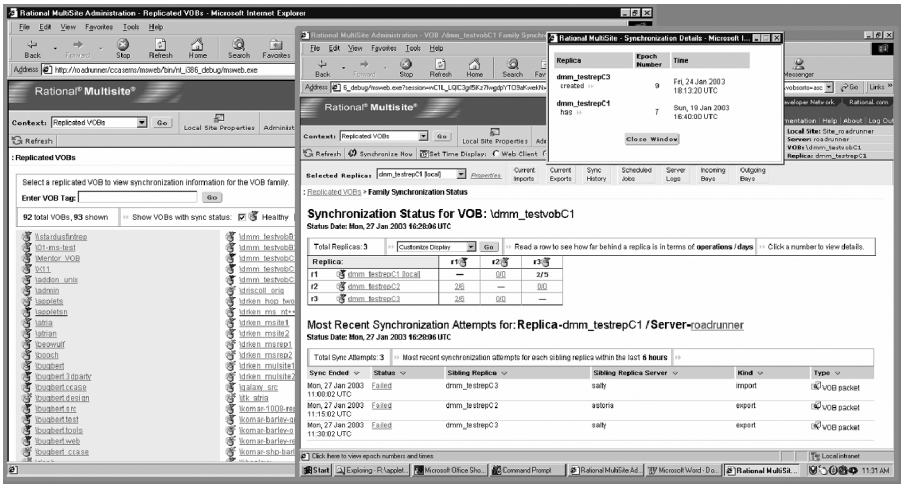


Figura 9.10 Software Rational MultiSite da IBM para o gerenciamento de software altamente distribuído (cortesia da IBM Rational Division).

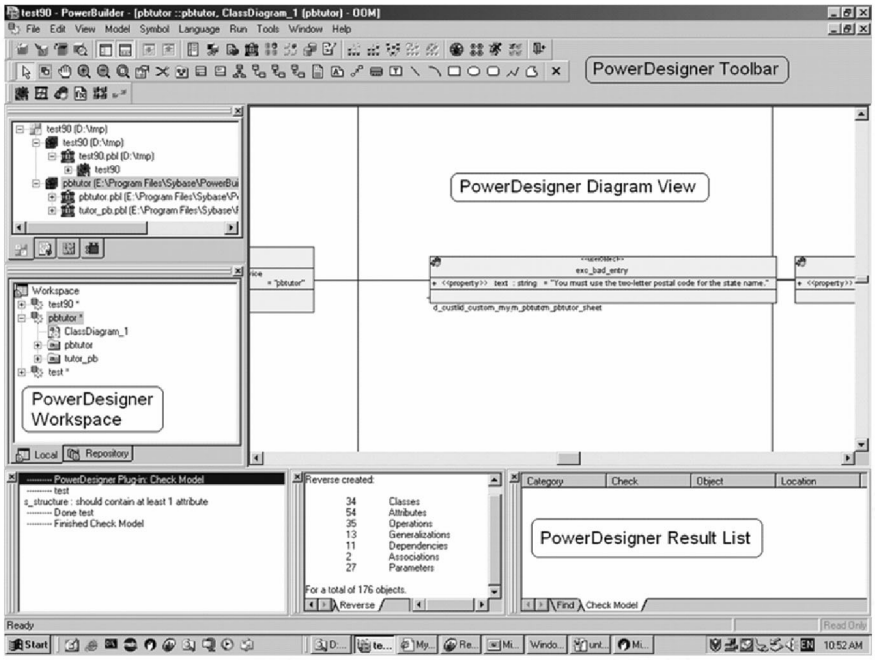


Figura 9.11 Plug-in do Sybase PowerDesigner para o Sybase PowerBuilder (figura de *http://www.pbugg.de/docs/1*, *Berndt Hambock*)

formas de desenvolvimento de software têm trabalhado para desenvolver essa plataforma durante as duas últimas décadas. O desafio é relativo pois simplesmente é difícil fazê-lo bem. As três empresas que estamos discutindo aqui possuem ofertas amplas e oferecem algum grau de integração com sua tecnologia CASE de projeto de banco de dados.

Para a Computer Associates, a marca AllFusion é uma família de ferramentas de ciclos de vida de desenvolvimento. Ela serve para abordar o projeto, a criação, a implantação e o gerenciamento de aplicações em negócios eletrônicos — eBusiness. O Sybase também tem um grande pacote de produtos, e o seu ponto forte está na tecnologia colaborativa. Do ponto de vista de projeto, a capacidade de conectar o Sybase PowerDesigner às suas ferramentas de desenvolvimento de aplicação populares, o Sybase Power Builder, é um recurso muito interessante, como vemos na Figura 9.11. As ferramentas da IBM são baseadas, mas não exclusivamente, nos seus produtos Rational, abrange tudo, desde a criação de requisitos até o gerenciamento de pacotes, controle de código-fonte, restrições de projeto arquitetônico, teste automatizado, análise de desempenho e desenvolvimento cross site . Uma representação da Software Development Platform aparece na Figura 9.12.

9.9 Verificação de compatibilidade de projeto

Como em todos os projetos complexos, e particularmente quando vários projetistas estão envolvidos, pode ser muito difícil manter a integridade do projeto do sistema. Os melhores arquitetos e projetistas de software lutam com isso definindo diretrizes e regras de projeto. Estas às vezes são chamadas de "padrões de projeto" e "antipadrões". Um padrão de projeto é um princípio de projeto ao qual geralmente devemos aderir dentro do projeto do sistema. Ao contrário, um antipadrão é exatamente o oposto. Ele representa falhas no projeto do sistema que podem ocorrer por meio de violação dos padrões de projeto ou pela definição explícita de um antipadrão. A imposição de padrões de projeto e antipadrões é um atributo emergente nas melhores ferramentas CASE para o projeto de sistemas, em geral, e para o projeto de banco de dados, em particular. A Figura 9.13 mostra um exemplo da interface utilizada no Rational Data Architect para verificação de compatibilidade, com varreduras do sistema para impor padrões de projeto e verificar antipadrões. Também existe algum grau de apoio ao padrão de projeto e antipadrão no AllFusion ERwin Data Modeler e no Sybase PowerDesigner. A verificação de compatibilidade nos produtos Rational da IBM é a mais madura em geral,