Indicadores de qualidade de uma implementação de banco de dados relacional

Glauco Galves Bordin

DISSERTAÇÃO APRESENTADA
AO
INSTITUTO DE MATEMÁTICA E ESTATÍSTICA
DA
UNIVERSIDADE DE SÃO PAULO

PARA OBTENÇÃO DO GRAU DE MESTRE
EM
CIÊNCIA DA COMPUTAÇÃO

Área de Concentração: **Ciência da Computação**
Orientador: **Prof. Dr. Marcelo Finger**

- São Paulo, julho de 2001 -

Indicadores de qualidade de uma implementação de banco de dados relacional

Este exemplar corresponde à redação final da dissertação devidamente corrigida e apresentada por Glauco Galves Bordin e aprovada pela Comissão Julgadora.

São Paulo, 02 de julho de 2001.

Banca Examinadora:

- Prof. Dr. Marcelo Finger (orientador) – IME-USP
- Prof. Dr. Francisco Carlos da Rocha Reverbel – IME-USP
- Prof. Dr. Caetano Traina Júnior – ICMC-USP

Indicadores de qualidade de uma implementação de banco de dados relacional

por Glauco Galves Bordin

Resumo

Esta tese propõe alguns indicadores para avaliação da qualidade de uma implementação de banco de dados relacional. Conceitos existentes na teoria do modelo relacional de banco de dados são utilizados como indicadores da qualidade de sua implementação, como é o caso das formas normais, das restrições de integridade entre outras. No entanto, neste trabalho serão propostos indicadores de qualidade com base em outros aspectos, como a ocorrência de sinônimos e homônimos, a ocorrência de divergências de domínio de campos, a ausência de informações de segurança e rastreabilidade, o mapeamento do modelo entidade-relacionamento para o modelo relacional, etc. É realizado ainda um estudo de casos com algumas implementações de sistemas reais, onde são apresentadas algumas conclusões tanto individuais de cada sistema, como comparativas.

Quality indicators of a relational database implementation

by Glauco Galves Bordin

Abstract

This thesis proposes a few indicators for the quality evaluation of a relational database implementation. Some concepts of the theory of relational database model are used as quality indicator like the normal forms, the integrity restrictions and others. This work proposes quality indicators based on others features, like the occurence of the synonymous and the homonymous, the occurrence of the field domain divergence, the absence of security and trace information, the mapping of the entity-relationship model to the relational model, etc. A case study is performed over the implementation of real systems, where some conclusions refering to individual systems as well as to comparisons between them are presented.

"Se diante de mim, não se abrir, o mar,
Deus me fará andar por sobre as águas"

[Cântico *Rompendo em Fé*]

Dedico este trabalho a Deus, por conceder a mim a graça de poder estudar, mesmo sendo este, um país onde tantas coisas faltam para muitos, aos meus pais pelo apoio e exemplo tanto nos estudos como na vida, e a minha esposa pelo amor e apoio com que sempre me auxiliou.

Agradecimentos

Ao professor Marcelo Finger, pela amizade, por orientar-me neste trabalho e pelas cobranças muitas vezes necessárias.

À minha família, pelo apoio e incentivo, principalmente dos meus pais. Em especial ao meu pai pela correção ortográfica deste trabalho.

À minha namorada, noiva e depois esposa, pela marcação cerrada que realizou com muito carinho e amor.

Ao Sérgio Luiz e à Paula Eugênio, meus chefes, por me darem a oportunidade de estudar e se desdobrarem com os meus horários confusos.

A todos os professores e amigos do IME com quem estudei tanto na graduação como na pós.

In Memoriam ao colega Carlos Daniel Chacur Alves, pelo companheirismo e incentivo.

A todos os funcionários do IME, que com o seu trabalho contribuíram direta ou indiretamente para a realização deste estudo.

Índice

Índice de Figuras

Capítulo 1

Introdução

"Qualquer programa faz alguma coisa certa, pode é não ser aquilo que se pretende" [Pressman 95].

Neste capítulo, será definida a proposta de estudo e sua motivação, sendo comentados alguns tópicos e características de assuntos relacionados, que serão discutidos de modo mais detalhado nos capítulos seguintes.

Os principais assuntos abordados neste trabalho são Teoria de Banco de Dados e Engenharia de Software. Enquanto o primeiro é responsável por fornecer o conhecimento técnico necessário para a obtenção dos dados, informações e aspectos que serão tratados e avaliados; o segundo se encontra relacionado à análise da qualidade da implementação de um banco de dados, sobre aspectos como flexibilidade, facilidade de manutenção, reutilização, entre outros que serão discutidos adiante.

1.1 Objetivo

O objetivo deste trabalho é estabelecer critérios de avaliação da qualidade de uma implementação de banco de dados relacional. Avalia-se a qualidade do produto, a implementação do banco de dados relacional, e não o processo de desenvolvimento do mesmo. Os critérios propostos para avaliação da qualidade são baseados em indicadores de qualidade que serão definidos no capítulo 5. Estes indicadores de qualidade consideram aspectos tanto do modelo relacional de banco de dados, como do mapeamento do modelo entidade-relacionamento para o relacional. Por ser avaliada a qualidade do produto já implementado, é necessário realizar uma engenharia reversa para obter algumas informações e características do modelo relacional.

Engenharia reversa é um tema que sempre desperta curiosidade e interesse, sendo adotada quando se deseja obter um entendimento em nível mais elevado de um sistema ou modelo. Isto é necessário, principalmente, ao se reescrever um sistema ou quando não se possui documentação e se deseja realizar a exportação, migração ou compartilhamento das informações contidas no sistema.

De maneira geral, o estudo dos processos diretos, como o desenvolvimento de sistemas, é mais comum e difundido. No caso do projeto de banco de dados relacionais, há uma teoria bastante conhecida e empregada para seu desenvolvimento, como é o caso das formas normais, da BCNF, do modelo entidade-relacionamento, etc. Através dos conceitos desta teoria, deseja-se ter um controle maior sobre a implementação do sistema, ou seja, sobre a qualidade da implementação, como, por exemplo, o armazenamento de informações sem redundância desnecessária e a recuperação de modo eficiente e consistente.

Para este trabalho, assume-se a inexistência de documentação sobre a implementação, sendo todas as informações extraídas diretamente do modelo de dados. No entanto, no passo seguinte do processo de avaliação proposto, onde é necessário um conhecimento do modelo de dados, a existência de documentação facilita essa tarefa. Visando compensar esta possível dificuldade, principalmente em sistemas grandes, foi desenvolvida uma ferramenta de auxílio ao agrupamento de campos segundo seu significado. Detalhes deste processo e da ferramenta serão discutidos no capítulo 7.

A aplicabilidade deste trabalho é extensa em sistemas de informação reais, ao quantificar e estabelecer a qualidade com que um produto foi desenvolvido e, em casos de manutenção de sistemas, verificar se as alterações são claras e seguem os padrões de implementação originais do sistema. Além disso, no caso de inexistência de documentação, uma vez realizado o processo proposto, tem-se mapeadas as informações utilizadas pelo sistema.

De posse do mapeamento das informações é possível identificar suas dependências, realizar alterações mais precisas no sistema, evitando erros conceituais e identificando a origem correta de dados para sua extração ou utilização em outros sistemas. Para validar os indicadores de qualidade propostos neste trabalho, foram realizados quatro estudos de casos, apresentados no capítulo 8.

Como será discutido no próximo item, a qualidade com que um software foi desenvolvido pode ser adotada como critério de avaliação de fornecedores de serviço, como requisitos contratuais impostos pelos clientes, como um diferencial mercadológico do fornecedor, etc.

1.2 Qualidade

Qualidade pode ser definida de diversas maneiras, em cada uma a ênfase dada varia entre produtos e serviços, clientes, processos etc., isto devido sua vasta aplicabilidade.

Estamos interessados na qualidade segundo o enfoque no produto, a implementação de banco de dados relacional. Neste sentido, a ISO (International Standard Organization) define *qualidade* como o conjunto das propriedades e características de um produto, processo ou serviço, que fornecem a capacidade de satisfazer as necessidades explícitas e implícitas.

Mais abaixo serão definidas algumas necessidades e características, segundo a Associação Brasileira de Normas Técnicas (ABNT), utilizadas para avaliar a qualidade de software.

Qualidade encontra-se presente em quase todas as corporações, sendo um item de avaliação utilizado por clientes, consumidores e até mesmo pelo próprio produtor. As principais razões que levam a adotá-la são:

- Requisitos contratuais, imposição dos clientes e consumidores;
- Melhoria da qualidade de serviços e produtos;
- Aumento da produtividade e melhoria de processos;
- Instrumento mercadológico com utilização em marketing;
- Ampliação de mercado consumidor, adotando padrões para exportação.

O objetivo da adoção de qualidade em software é a redução dos custos e tornar o produto mais competitivo.

Para ser gerenciada, a qualidade necessita ser medida de maneira a evidenciar o grau com que os objetivos de sua definição são atendidos e/ou excedidos. As medidas de qualidade determinam o estado atual dos processos, produtos, serviços etc. e provêm um parâmetro de avaliação para mudanças e melhorias. Estas medidas são apresentadas neste estudo sob a forma de indicadores de qualidade definidos no capítulo 5.

Segundo a [NBR 13596/96], versão brasileira da ABNT para a ISO/IEC 9126, a qualidade de software apresenta as seguintes características e sub-características:

- *Funcionalidade* - evidencia que o conjunto de funções atendem às necessidades explícitas e implícitas para a finalidade a que se destina o produto;
    - *Adequação* – presença de conjunto de funções e sua apropriação para as tarefas;
    - *Acurácia*[1] – geração de resultados ou efeitos corretos;
    - *Interoperabilidade* – Capacidade de interagir com outros sistemas;
    - *Conformidade* – estar de acordo com normas, convenções, regulamentações;
    - *Segurança de Acesso* – capacidade de evitar acesso não autorizado a programas e dados;

- *Confiabilidade* – evidencia que o desempenho se mantém ao longo do tempo e em condições estabelecidas;
    - *Maturidade* – freqüência de falhas;
    - *Tolerância a Falhas* – manter nível de desempenho em caso de falhas;
    - *Recuperabilidade* – capacidade de se restabelecer e restaurar dados após falha;

- *Usabilidade* – evidencia a facilidade para utilização do software
    - *Inteligibilidade* – facilidade de entendimento dos conceitos utilizados;
    - *Apreensibilidade* – facilidade de aprendizado;
    - *Operacionalidade* – facilidade de operar e controlar a operação;

- *Eficiência* - evidencia que os recursos e os tempos envolvidos são compatíveis com o nível de desempenho requerido para o produto;
    - *Comportamento em Relação ao Tempo* – tempo de resposta, de processamento;
    - *Comportamento em Relação a Recursos* – quantidade de recursos utilizados;

1 N.A.: O original da NBR 13596 utiliza o termo *Acurácia*, embora o melhor termo para esta sub-característica seja *Precisão*, é encontrada na ISO/IEC 9126.

- *Manutenibilidade* – evidencia que há facilidade para correções, atualizações e alterações;
    - *Analisabilidade* – facilidade de diagnosticar deficiências e causas de falhas;
    - *Modificabilidade* – facilidade de modificação e remoção de defeitos;
    - *Estabilidade* – ausência de riscos de defeitos inesperados;
    - *Testabilidade* – facilidade de ser testado;

- *Portabilidade* – evidencia que é possível utilizar o produto em diversas plataformas com pequeno esforço de adaptação;
    - *Adaptabilidade* – capacidade de ser adaptado a ambientes diferentes;
    - *Capacidade para ser Instalado* – facilidade de instalação;
    - *Conformidade* – acordo com padrões ou convenções de portabilidade;
    - *Capacidade para Substituir* – substituir outro software;

No entanto, algumas características e/ou sub-características que agregam valor na avaliação da qualidade de um software não se encontram presentes nas definições de características da NBR 13596, como é o caso de:

- *Reutilização* - corresponde ao grau com que um sistema ou parte deste pode ser utilizada em outros sistemas [Pressman 95];
- *Produtividade* - alcançada com o uso do sistema [Pressman 95].

Neste estudo, estarão sendo avaliadas algumas destas características de qualidade de software através da definição dos indicadores de qualidade, detalhados no capítulo 5.

1.3 Comparações com a Literatura

Este trabalho utiliza principalmente conceitos de banco de dados e seu desenvolvimento, a este respeito a literatura existente é bastante comum e conhecida, como é o caso de [Korth e Silberschatz 95], [Ullman 88], [Elmasri e Navathe 89], [Batini, Ceri e Navathe 92], [Setzer 89], [Heuser 99], [Date 86] e [Mesquita 98]. Abordando o processo de desenvolvimento de banco de dados, modelagem entidade-relacionamento, modelo relacional, formas normais, mapeamento do modelo entidade-relacionamento para o relacional, além de outros assuntos fundamentados na teoria proposta por [Chen 76] e [Codd 70].

Por outro lado, são empregados também conceitos de engenharia de software e qualidade como: os principais modelos de desenvolvimento de software, suas vantagens e desvantagens, conceitos básico e um pouco da história da utilização de qualidade, entre outros. Sobre qualidade, o principal enfoque da discussão está no conceito de indicadores de qualidade, seu significado e utilização. Os assuntos de engenharia de software discutidos podem ser encontrados em [Pressman 95], [Boehm 98], [Gilb 93], [McCracken e Jacson 82], [Royce 70] e de qualidade em [Rolt 98], [Crosby 94], [Deming 90], [Feingenbaum 86], [Garvin 92], [Gouzze, Mazijn e Billharz 95], [Ishikawa 86], [Juran 92] e [Oliveira, Latelme e Formoso 95].

Os indicadores propostos para avaliação da qualidade de uma implementação de banco de dados relacional e sua aplicação em casos reais, são as principais contribuições deste trabalho. É possível ainda, adotar os indicadores de qualidade para realizar uma análise comparativa da qualidade de diferentes implementações, inclusive para implementações de tamanhos diferentes.

Estes indicadores de qualidade são definidos a partir de valores quantitativos de itens presentes no modelo relacional e do mapeamento do modelo entidade-relacionamento para o relacional. Como é assumido que a única fonte de informação disponível para extração destes valores é a própria implementação do banco de dados, é necessário a utilização de algumas técnicas de engenharia reversa, que podem ser encontrados em [Andersson 94], [Premerlani e Blaha 94], [Heuser 99], [Freitas e Leite 97] e [Neild e Henschen 97]. Em geral, a literatura a este respeito, trata do caso em que se deseja realizar a engenharia reversa de um modelo relacional de banco de dados, visando obter um modelo equivalente orientado a objetos, ao contrário do objetivo de seu emprego neste trabalho, que é o de fornecer os valores para determinação dos indicadores de qualidade definidos.

O resultado do processo de engenharia reversa sobre a implementação analisada, gera um mapeamento das informações do sistema que pode ser utilizado em projetos como data warehouse, data mart, extração e compartilhamento de dados etc.

A utilização de engenharia reversa como ferramenta de análise da qualidade de uma implementação de banco de dados foi apenas citada em [DAMA 98], onde este processo foi definido por "Data Profiling". A maioria dos tópicos que compõem a definição de "Data Profiling", como: a análise de sinônimos e homônimos, a análise de dependências e avaliação de qualidade; estão contidos nos indicadores propostos por este trabalho.

1.4 Organização da Tese

Este trabalho está dividido em 10 capítulos, dos quais a Introdução é o primeiro.

No capítulo 2, "Projeto de Banco de Dados", é definido o processo de projeto ideal de um banco de dados, quais os níveis envolvidos e modelos resultantes de cada um. Em seguida, no capítulo 3, "O Modelo Entidade – Relacionamento", são apresentados os elementos fundamentais deste modelo do nível conceitual. O Modelo Lógico Relacional é apresentado no capítulo 4.

Até então, o conteúdo possui literatura bastante ampla e estudada, como foi descrito no item anterior. No capítulo 5, são introduzidos alguns conceitos de qualidade e propostos alguns indicadores para o estudo, que é o de mensurar e permitir a comparação da qualidade de implementação entre banco de dados relacionais. Para isto, faz-se necessária uma análise das implementações dos modelos lógicos dos projetos de bancos de dados, este tema é abordado no capítulo 6, "Técnicas de Engenharia Reversa".

As funcionalidades e detalhes de implementação da ferramenta desenvolvida para extrair os dados necessários para obtenção dos valores dos indicadores e auxiliar a análise da qualidade das implementações, são descritas no capítulo 7, "Implementação de Ferramenta de Auxílio à Análise de Qualidade".

No capítulo 8, "Estudo de Casos", são apresentados alguns casos, ou seja, exemplos reais, em que a ferramenta e a metodologia proposta são aplicadas e, a partir dos resultados individuais obtidos, é realizada a análise de cada caso.

O capítulo 9, "Conclusões", apresenta as conclusões obtidas do processo, da ferramenta, de sua utilização e implementação, da análise comparativa dos casos analisados e do trabalho realizado. Por fim, na "Bibliografia", encontram-se as referências bibliográficas utilizadas.

Capítulo 2

Projeto de Banco de Dados

"Processo em que se deseja representar informações do mundo real na forma de dados armazenados em estruturas e manipulados de forma automatizada por um sistema computacional" [Heuser 99].

Neste capítulo, serão fornecidas as idéias e os conceitos necessários para o processo de desenvolvimento de um sistema de banco de dados: quais as fases envolvidas no projeto e os produtos resultantes de cada fase. Será realizada ainda, uma comparação entre o processo ideal de desenvolvimento de um sistema de banco de dados e os principais modelos de desenvolvimento de software presentes na literatura de engenharia de software. Será ainda discutido, o processo de engenharia reversa de um sistema de banco de dados, cujas principais abordagens e técnicas serão vistas no capítulo 6.

Os conceitos e processos discutidos neste capítulo serão utilizados na identificação do escopo e objetivo deste trabalho em comparação com o processo ideal de desenvolvimento de um banco de dados.

2.1 Processo de Desenvolvimento de Sistema de Banco de Dados

Segundo [Korth e Silberschatz 95], um *sistema de banco de dados*, corresponde a um sistema projetado para gerenciar grandes grupos de informação, envolvendo a definição de estruturas para armazenamento de informação e mecanismos para manipulá-las, além de fornecer segurança das informações armazenadas.

O *projeto de banco de dados* corresponde à atividade que leva à implementação de um sistema de banco de dados. Em geral, este processo é dividido em três etapas ou projetos: *Projeto Conceitual*, *Projeto Lógico* e *Projeto Físico*. Como produto resultante de cada etapa ou projeto, é gerado um modelo de dados, respectivamente: *Modelo Conceitual*, *Modelo Lógico* e *Modelo Físico*.

O *modelo de dados* é uma coleção de ferramentas conceituais utilizadas para descrever, de modo formal, a representação das informações de um sistema como dados e suas semânticas, restrições e relacionamentos.

Na figura a seguir, é representado o processo ideal de projeto de banco de dados, que conforme será visto no item 2.7, é baseado no modelo em cascata de desenvolvimento de software:

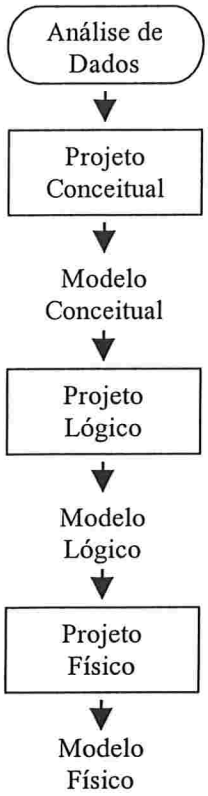


Fig. 2.1 – *Processo Ideal de Projeto de Banco de Dados*

A seguir serão introduzidos cada um dos elementos do projeto de um banco de dados. Estes elementos podem ser encontrados em [Korth e Silberschatz 95], [Mesquita 98], [Setzer 89], [Heuser 99], [Ullman 88] e [Date 86].

2.1.1 Projeto Conceitual

O *projeto conceitual*, também chamado *modelagem conceitual*, é o processo que parte da análise de requisitos e da observação de fatos, ações e elementos relevantes da realidade, com o objetivo de produzir um modelo de dados conceitual correspondente a uma representação da realidade em alto nível de abstração, onde estarão contidas as necessidades em termos de informação para o sistema, independente de sua implementação.

2.1.2 Modelo Conceitual

Produto final da primeira etapa do projeto de sistema de banco de dados, o *modelo conceitual* determina quais informações do *universo de discurso*, devem ser representadas no sistema e quais as relações existentes entre elas, independente da implementação.

Existem vários modelos de dados conceituais como o orientado a objetos e o entidade-relacionamento.

O *modelo entidade-relacionamento* possui larga utilização em projetos de sistemas de banco de dados, e é baseado no conceito de objetos básicos chamados entidades e em seus relacionamentos.

O *modelo orientado a objetos* possui muitos conceitos do modelo entidade-relacionamento, mas é baseado no encapsulamento de código e dados em uma única unidade chamada *objeto*.

2.1.3 Projeto Lógico

O objetivo do *projeto lógico* é o de transformar a representação obtida pelo modelo conceitual, em uma especificação que descreva as estruturas que poderão ser implementadas por um sistema de computação, mas sem considerar nenhuma característica específica de um sistema gerenciador de banco de dados ou de uma linguagem.

O produto resultante deste processo é o modelo lógico, ou esquema lógico, do banco de dados.

2.1.4 Modelo Lógico

Produto resultante do projeto lógico, o *modelo lógico* descreve as estruturas a serem implementadas no banco de dados, de forma independente de sistema gerenciador de banco de dados e de linguagem.

Os modelos lógicos podem ser divididos em dois tipos: os *modelos lógicos baseados em registros*, que recebem este nome por serem estruturados em registros de formato fixo de diversos tipos e, ao contrário dos modelos baseados em objetos, não incluem nenhuma representação direta para código dentro do banco de dados; e *os modelos baseados em objetos*, que apresentam estruturas mais complexas que as do modelo baseado em registros e permitem a definição de operações de manipulação dos dados sem limite de tipo de dados ou linguagem de consulta.

Os principais modelos lógicos baseados em registros são: o modelo de Redes, o Hierárquico e o Relacional.

O *modelo de redes* é baseado em dados armazenados em registros e ligações representadas por ponteiros destes registros a outros registros, resultando numa organização de grafos arbitrários.

O *modelo hierárquico*, a exemplo do modelo de redes, é baseado em registros e ligações, diferindo no fato de que os registros são armazenados como coleções de árvores, ao invés de grafos arbitrários.

O *modelo relacional* é o mais utilizado em sistemas de banco de dados, representando os dados e relacionamentos como um conjunto de tabelas, cada uma composta por um conjunto de colunas.

Os primeiros sistemas de banco de dados basearam-se no modelo de redes ou no modelo hierárquico, este últimos modelos são mais dependentes da implementação do banco de dados que o modelo relacional. O modelo relacional permitiu que fosse desenvolvida uma teoria substancial que auxilia o projeto e o processamento eficiente de consultas em um banco de dados relacional.

Como exemplo de modelo lógico baseados em objetos temos: o modelo objeto-relacional e o modelo de objetos.

O *modelo orientado a objetos* se caracteriza por fornecer de forma conveniente, capacidade de estruturação flexível e admitir restrições de dados para serem explicitamente especificados. O modelo orientado a objetos é baseado em um conjunto de objetos. Um objeto possui também trechos de código que operam sobre o objeto.

O *modelo objeto-relacional* corresponde a uma extensão do modelo relacional fornecendo tipos de dados mais flexíveis, através da inclusão de características de orientação a objetos, e construções de linguagem de consulta relacional específicas para a manipulação destes tipos de dados.

2.1.5 Projeto Físico

Por *projeto físico* entende-se o processo que visa traduzir o modelo lógico para uma descrição detalhada e em baixo nível, das estruturas de armazenamento e mecanismos de acesso eficientes ao conteúdo do banco de dados, levando-se em consideração, características e aspectos da linguagem ou sistema gerenciador de banco de dados utilizados para implementar o sistema.

2.1.6 Modelo Físico

O *modelo físico* define as estruturas físicas de armazenamento de dados, tais como: índices, arquivos de dados, clustering, hashing, árvores-B, etc., projetados a fim de obter-se melhor processamento e uso mais econômico dos recursos.

2.2 O Processo de Desenvolvimento

O modelo de desenvolvimento de projeto de banco de dados acima apresentado, é baseado no mesmo princípio que rege o modelo de ciclo de vida do desenvolvimento tradicional de sistemas, o *modelo em cascata* ou *"waterfall model"*, definido por Royce em 1970 [Royce 70].

Este modelo apresenta passos bem definidos e seqüenciais, cada um obtendo um produto como resultado que é utilizado na fase seguinte. No caso, o modelo proposto para o desenvolvimento de banco de dados parte do nível mais abstrato com objetos, características e comportamentos do mundo real e, gradativamente através do processo de desenvolvimento, passa-se para níveis de menor abstração até obter a implementação do sistema.

O modelo em cascata pressupõe que ao passar para a fase seguinte, a anterior estará concluída, ou seja, permanecerá congelada durante todo o restante do ciclo de vida do processo de desenvolvimento, até o seu final.

Na prática, o processo de desenvolvimento de um sistema de informação costuma ser menos estático e comportado do que o proposto, como é apresentado na figura 2.2, sendo sujeito a alterações e mudanças em fases anteriores, ou seja, a voltas para os projetos e modelos de nível mais alto, ou o que é pior, assumir mudanças e alterações de um nível para baixo, sem atualizar os níveis superiores, gerando inconsistência entre eles.

Estas atitudes indesejadas, mais freqüentes em manutenções e correções de erros do sistema após sua implantação, costumam ocasionar problemas como inconsistência entre os modelos, novos erros nas aplicações, modelos desatualizados, problemas de performance e aumento da complexidade de entendimento do sistema.

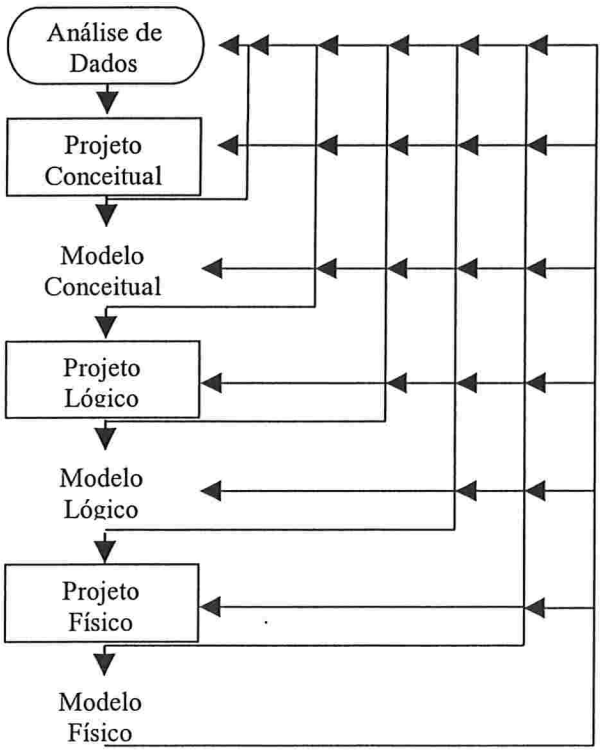


Fig. 2.2 – *Projeto de Banco de Dados*

Para expressar este dinamismo necessário e constante no processo de desenvolvimento de um sistema, outros modelos de ciclo de vida de desenvolvimento de sistemas surgiram, muitos dos quais derivados do modelo em cascata, como é o caso do *modelo evolucionário*, proposto por McCracken e Jackson em 1982 [McCracken e Jacson 82], e o *modelo incremental*, destacado por Tom Gilb [Gilb 93].

Basicamente, estes modelos correspondem à adoção do modelo em cascata em sucessivas expansões incrementais do sistema desenvolvido. Como serão expandidas as partes do sistema já implementadas, é necessário que estas tenham sido implementadas com qualidade, a fim de serem flexíveis e fáceis de comportarem alterações.

O *modelo em espiral*, proposto por Boehm em 1988 [Boehm 88], presente na figura 2.3, assume que o processo de desenvolvimento ocorre em ciclos, cada um apresentando etapas que envolvem elaboração, avaliação, desenvolvimento e planejamento do ciclo seguinte. No entanto, na execução das tarefas de cada ciclo, podem ser utilizados outros modelos de ciclo de vida, como a utilização do modelo em cascata na implementação de sistema, prototipagem para avaliação de riscos, elaboração de requisitos, etc.

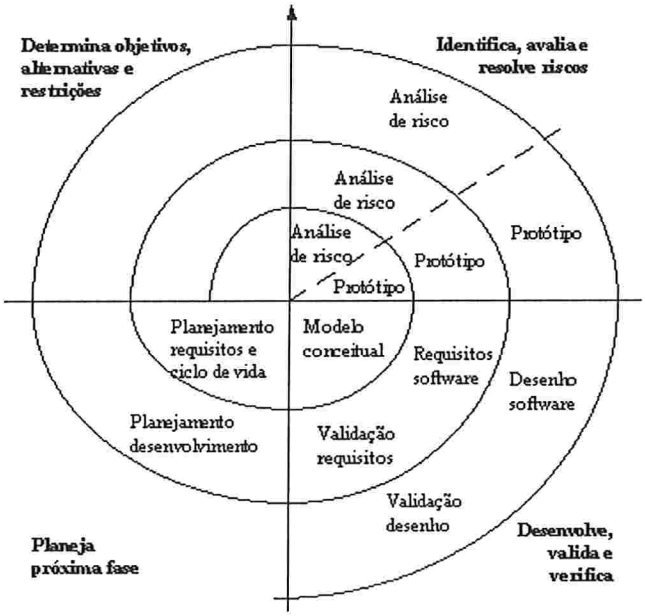


Fig. 2.3 – *Modelo de desenvolvimento de sistemas em espiral*

Como o modelo em espiral é mais complexo, requer uma capacidade maior de gestão, pois permite o desenvolvimento de tarefas em paralelo tornando mais difícil a determinação de prazos, custos e indicadores de progresso.

O resultado das características indesejadas do processo de desenvolvimento tradicional, como o fato de se supor que o sistema seja estático e, do contrário, de prováveis inconsistências entre as fases; pode significar que as soluções adotadas para os problemas

nem sempre são as melhores ou foram realizadas da melhor maneira. Portanto a consistência entre os níveis do projeto fica comprometida, causando uma queda: na *qualidade do sistema*, ou seja , das funcionalidades e objetivos para os quais o uso do sistema se propõe; na *qualidade do processo de desenvolvimento do sistemas*, ou seja, o processo de projeto e desenvolvimento do sistema não foi realizado com qualidade, pois as documentações, modelos dos sistemas etc., não estão consistentes; e na *qualidade da implementação do sistema*, ou seja, do funcionamento, da facilidade de manutenção e entendimento da implementação do sistema.

Pode-se considerar a qualidade do sistema: o impacto do uso do sistema no mundo real; a qualidade do processo de desenvolvimento do sistema: o impacto no processo de geração e alteração do projeto do sistema; e qualidade da implementação do sistema: o impacto na forma como o projeto do sistema se concretizou, ou seja, como foi implementado.

Neste trabalho, vamos nos ater ao estudo deste último item, a qualidade da implementação do sistema, que como foi visto anteriormente, tem papel fundamental nos modelos de desenvolvimento de sistemas, uma vez que são sujeitos a expansões e alterações. Para isto serão definidos, no capítulo 5, indicadores de qualidade que permitam medir a qualidade e analisar a implementação de um sistema de banco de dados. Estes indicadores, ao mesmo tempo, podem ser utilizados de uma maneira pró-ativa para identificar e determinar tópicos e ocorrências de casos na implementação do sistema que podem ser corrigidos a fim de se melhorar a qualidade da implementação do sistema.

Para obter estes indicadores, faz-se necessário o emprego de técnicas de engenharia reversa, presentes em maiores detalhes no capítulo 6, pois é assumido que o modelo do sistema não existe ou não é confiável, o que ocorre com bastante freqüência em sistemas reais.

2.3 Engenharia Reversa

Engenharia Reversa corresponde ao termo atribuído sempre que se realiza um processo que corresponde ao inverso do processo normalmente realizado [Heuser 99]. No caso de desenvolvimento de banco de dados, parte-se de uma implementação de um sistema, o *Modelo de Implementação,* e resulta em uma especificação, o *Modelo Conceitual*, descrevendo abstratamente a implementação de origem.

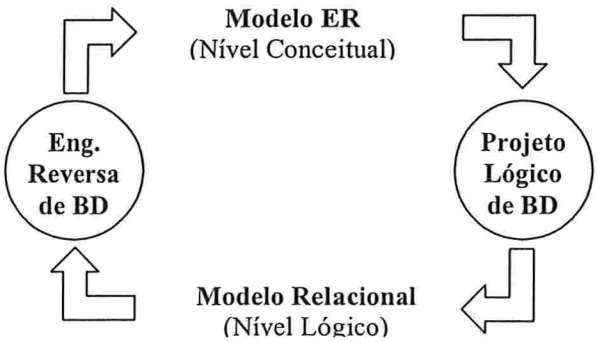


Fig. 2.4 – *Engenharia Reversa de BD*

O principal objetivo da engenharia reversa é o de compreender as implementações de software já desenvolvidas. Este conhecimento é necessário quando há interesse em expandir ou alterar um sistema legado que não possua documentação; quando ocorrer alguma alteração no sistema que não tenha sido registrada; no caso de migração de dados, manutenção, reengenharia ou qualquer outra tarefa que envolva o conhecimento a respeito do comportamento do software ou de seus dados.

A engenharia reversa representa um problema de grande complexidade computacional, onde é constante a necessidade de interagir com seres humanos para complementar lacunas de conhecimento e tomadas de decisão. Por outro lado, é imprescindível o auxílio de ferramentas computacionais para a realização desta tarefa [Freitas e Leite 97].

O tema engenharia reversa de banco de dados será discutido com maiores detalhes no capítulo 6, mas pode ser encontrado também em [Heuser 99], [Andersson 94], [Premerlani e Blaha 94] e [Freitas e Leite 97].

Nos capítulos seguintes serão explorados os modelos entidade-eelacionamento, pertencente ao nível conceitual do projeto de um banco de dados, e o modelo relacional, pertencente ao nível lógico. Ambos serão utilizados como base conceitual para a formulação dos indicadores de qualidade de uma implementação de banco de dados relacional. Além de expostos os conceitos de cada um dos modelos, será discutido também, o mapeamento do modelo entidade-relacionamento para o modelo relacional. Este mapeamento é útil tanto no processo direto de desenvolvimento como no processo de engenharia reversa.

Capítulo 3

O Modelo Entidade-Relacionamento

"O modelo de dados entidade-relacionamento baseia-se na percepção de um universo constituído por um grupo básico de objetos chamados entidades e por relacionamentos entre estes objetos" [Korth e Silberschatz 95].

O objetivo deste capítulo é introduzir os conceitos básicos do modelo entidade-relacionamento, que serão adotados tanto no capítulo 4, quando será exposto o modelo relacional e o mapeamento do modelo entidade-relacionamento para o relacional, como no capítulo 5, quando serão definidos os indicadores de qualidade, alguns dos quais baseados em conceitos do modelo entidade-relacionamento.

O *Modelo Entidade-Relacionamento* é o modelo de dados mais adotado em projetos conceituais de banco de dados. Definido originalmente por Peter Chen em 1976 [Chen 76], tem por base a teoria relacional criada por E. F. Codd em 1970 [Codd 70].

Inicialmente, este modelo era constituído apenas pelos conceitos de objetos básicos como: entidades, relacionamentos e atributos; posteriormente, outros conceitos foram sendo agregados, como é o caso de: atributos chaves, generalizações etc. Estes conceitos agregados, podem ser encontrados em [Elmasri e Navathe 89], [Korth e Silberschatz 95], [Setzer 89], além de outros.

Os elementos básicos do modelo entidade-relacionamento proposto por Peter Chen serão discutidos a seguir.

3.1 Entidades

Entidade corresponde a uma representação de um objeto do mundo real de modo a identificá-lo de forma independente e única no universo do objeto. O objeto pode pertencer ao universo concreto ou abstrato da realidade representada.

Um conjunto de entidades do mesmo tipo, ou seja, que possuam características semelhantes é chamado *conjunto de entidades*. Conjuntos de entidades não são necessariamente disjuntos, pois uma entidade de um conjunto de entidades pode pertencer a outros conjuntos de entidade.

Uma entidade é constituída por um conjunto de atributos, e para cada atributo existe um conjunto de valores permitidos, chamado de *domínio*. No modelo entidade-relacionamento, os conjuntos de entidades são representados graficamente através de retângulos, conforme exemplifica a figura 3.1.

Fig. 3.1 – *Conjunto de entidades Produto e Loja*

3.2 Relacionamentos

Um *relacionamento* corresponde à associação entre duas ou mais entidades. Formalmente, um relacionamento é uma relação matemática em $n \geq 2$ conjuntos de entidades. Se E_1, E_2, E_3, ..., E_n são conjuntos de entidades, então o conjunto de relacionamentos R é um subconjunto de $\{ (e_1,e_2,e_3,...,e_n) \,/\, e_1 \in E_1, e_2 \in E_2, e_3 \in E_3, ..., e_n \in E_n \}$, onde $(e_1,e_2,e_3,...,e_n)$ é um relacionamento. Ao conjunto das associações de um, dois ou mais conjuntos de entidades, dá-se o nome de *conjunto de relacionamentos*. Os conjuntos de relacionamentos são representados no modelo de entidade–relacionamento por meio de um losango, como mostra a figura 3.2.

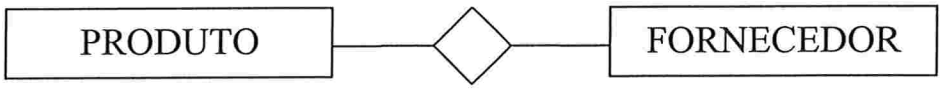


Fig. 3.2 – *Conjunto de relacionamentos Fornece entre os conjuntos de entidades Produto e Fornecedor*

O *grau de um relacionamento* corresponde ao número de entidades associadas num relacionamento. Relacionamentos de grau 2, como o do exemplo da figura 3.2, são chamados *binários*, de grau 3 são chamados *ternários*. Relacionamentos de grau n são chamados *n-ários*. Relacionamentos que associam entidades pertencentes ao mesmo conjunto de entidades recebem o nome de *auto-relacionamentos*.

À quantidade máxima e mínima de entidades de um conjunto de entidades em um relacionamento, dá-se o nome de *cardinalidade máxima e mínima*. Em relacionamentos binários, que correspondem aos mais freqüentes, a cardinalidade é classificada em:

Um-para-Um (1:1) - quando cada entidade de um conjunto de entidades A está associada, através do relacionamento, a no máximo uma entidade do outro conjunto de entidades B, com A e B não necessariamente distintos. Formalmente temos que um relacionamento $R \subseteq \{ (a_i,b_j) \,/\, a_i \in A \text{ e } b_j \in B \}$, com A e B conjuntos de entidades, é dito 1:1 se, e somente se, para $(a_k,b_m) \in R$, $(a_k,b_n) \in R$ e $(a_r,b_n) \in R$, $a_k = a_r$ e $b_m = b_n$.

Um-para-Muitos (1:n) - quando cada entidade de um conjunto de entidades A está associada, através do relacionamento, a qualquer número de entidades do outro conjunto de entidades B, mas cada entidade do conjunto de entidades B está associada a apenas uma entidade do conjunto de entidades A, com A e B não necessariamente distintos. Formalmente temos que um relacionamento $R \subseteq \{ (a_i,b_j) \,/\, a_i \in A \text{ e } b_j \in B \}$, com A e B conjuntos de entidades, é dito 1:n se, e somente se, para $(a_k,b_m) \in R$, $(a_r,b_m) \in R$, $a_k = a_r$.

Muitos-para-Muitos. (n:n) – quando cada entidade de um conjunto de entidades A está associado a qualquer número de entidades do conjunto de entidade B e cada entidade de B está associada a qualquer número de entidades de A, com A e B não necessariamente distintos. Formalmente temos que não há nenhuma restrição para este tipo de relacionamento, ou seja, quando um relacionamento $R \subseteq \{ (a_i,b_j) \,/\, a_i \in A \text{ e } b_j \in B \}$, com A e B conjuntos de entidades, não possui restrições de cardinalidade, é dito n:n.

A participação de cada um dos conjuntos de entidades em um conjunto de relacionamentos pode ser classificado como *total* ou *parcial*, é dita *total* quando toda entidade de um conjunto de entidades deve ser relacionada às entidades dos outros conjuntos de entidades do conjunto de relacionamentos; e é dita *parcial* quando apenas uma parte das entidades se relaciona com as entidades dos outros conjuntos de entidades.

Dependência de existência ocorre quando a existência de uma entidade x depende da existência de uma entidade y, neste caso x é dito dependente de existência de y. Operacionalmente, significa que se y deixar de existir, x também deverá ser eliminado. A entidade y é chamada *entidade dominante* e x *entidade subordinada*.

Um relacionamento pode possuir atributos descritivos, isto ocorre quando o atributo não é próprio de nenhuma das entidades associadas, mas sim do relacionamento entre elas, neste caso o atributo é chamado *atributo de relacionamento*.

3.3 Atributos

Atributos são propriedades que caracterizam e são particulares às entidades ou relacionamentos que descrevem. Formalmente, atributos são funções que aplicadas à entidade ou relacionamento, levam ao *domínio do atributo*, ou seja, um conjunto de valores válidos para o atributo. Portanto, toda entidade é descrita como um conjunto de pares (atributo, valor do domínio).

Atributos podem ser *mono-valorados*, quando representam um único valor, ou *multi-valorados*, quando representam mais de um valor.

Cada atributo está associado a um conjunto de valores válidos, que corresponde a um *domínio*.

Quando o atributo é definido sobre um único domínio, de modo que os valores que o atributo pode assumir sejam indivisíveis, o atributo é dito *atômico*.

Atributo global corresponde a um atributo cujo valor é o mesmo para todos elementos de um conjunto de entidades ou relacionamentos.

3.4 Agregação

Agregação é uma extensão do modelo entidade-relacionamento onde um conjunto de relacionamentos é representado como um conjunto de entidades, agregando todas informações relacionadas. No exemplo 3.3, é dado um conjunto de relacionamentos R_1 entre os conjuntos de entidades E_1 e E_2 que formam a agregação A que relaciona-se através do relacionamento R_2 com a entidade E_3.

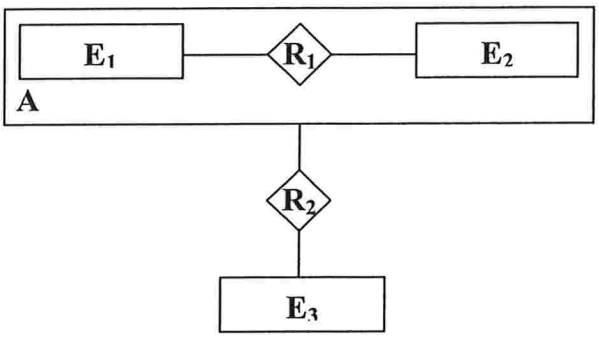


Fig.3.3 - *Agregação*

O conjunto de relacionamentos R_2 entre a o conjunto de entidades agregada A e o conjunto de entidade E_3, inclui os atributos que identificam as tuplas do conjunto de entidades E_1 e E_2, além dos atributos do conjunto de relacionamentos R_2.

Formalmente, $R_2 \subseteq \{ (a,e_3) \,/\, a \in A \text{ e } e_3 \in E_3 \} = \{ ((e_1,e_2),e_3) \,/\, (e_1,e_2) \in R_1 \text{ e } e_3 \in E_3 \}$.

3.5 Hierarquia de Generalização

Hierarquia de generalização corresponde à junção de dois ou mais conjuntos de entidades parcialmente distintos em um único conjunto de entidades. Formalmente, um conjunto de entidades E é uma *generalização* dos conjuntos de entidades E_1, E_2, ..., E_n se cada entidade dos conjuntos E_1, E_2, ..., E_n for também uma entidade de E.

Numa hierarquia de generalização, as propriedades do conjunto de entidades genérico, ou seja do conjunto de entidades básico, são herdados pelos conjuntos de entidades especializados, ou conjuntos de entidades filhos, isto significa que todo atributo, relacionamento ou generalizações que o conjunto de entidades genérico possui são herdados para as entidades especializadas.

A representação gráfica no modelo entidade-relacionamento para generalizações é um triângulo com uma seta incidindo no conjunto de entidades genérico e linhas unindo o triângulo a seus conjuntos de entidades especializados.

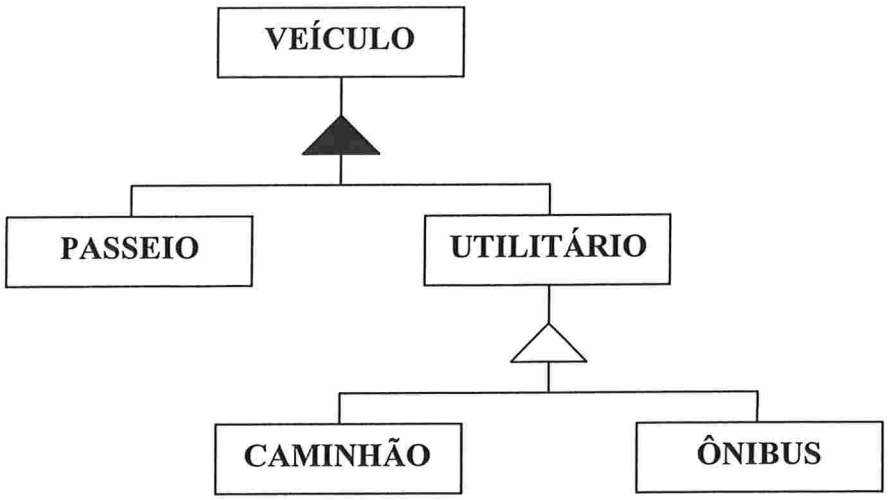


Fig.3.4 – *Hierarquia de Generalização*

Hierarquias de Generalização podem possuir *cobertura total* quando cada entidade do conjunto de entidades genérica é mapeada para uma entidade dos conjuntos de entidades especializadas, ou *cobertura parcial* quando existe ao menos uma entidade do conjunto de entidades genérica que não pode ser mapeada. Graficamente, quando o triângulo da generalização é preenchido, significa uma cobertura total; quando vazado, uma cobertura parcial.

Na figura 3.4, cada entidade do conjunto de entidades *Veículo* é necessariamente mapeada para uma entidade do conjunto de entidades *Passeio* ou *Utilitário*, no entanto, uma entidade do conjunto de entidades *Utilitário* não precisa necessariamente ser uma entidade dos conjuntos de entidades *Caminhão* ou *Ônibus*, podendo ser, por exemplo um furgão.

No próximo capítulo, será definido o modelo relacional e suas propriedades. Muitos dos conceitos do modelo entidade-relacionamento introduzidos neste capítulo serão utilizados, principalmente no mapeamento do modelo entidade-relacionamento para o modelo relacional.

Capítulo 4

O Modelo Relacional

"O modelo de dados relacional representa o banco de dados como uma coleção de tabelas" [Heuser 99].

A seguir, o modelo relacional de banco de dados será definido de maneira formal. Este modelo é amplamente empregado em sistemas comerciais. Serão também definidas neste capítulo, as características que determinam uma implementação de banco de dados com integridade, recuperabilidade de informação, sem redundâncias desnecessárias e inconsistências de dados. Faz parte dos objetivos deste capítulo a formalização do mapeamento do modelo entidade-relacionamento para o relacional, útil por exemplo, no processo de engenharia reversa do modelo relacional para o entidade-relacionamento, presente no capítulo 6.

O modelo relacional é um modelo lógico de banco de dados proposto por E. F. Codd [Codd 70], do Laboratório de Pesquisa da IBM de San Jose, Califórnia, no final da década de 60. Inicialmente denominado por System R, definia a organização dos dados e as linguagens formais para sua manipulação. Com base nessas linguagens formais, foi definida a primeira versão da linguagem SQL, Structured Query Language. Desde então, o modelo relacional estabeleceu-se como principal modelo de dados adotado por sistemas e a linguagem SQL como padrão de linguagem de gerenciamento de banco de dados relacionais.

Este modelo representa os dados na forma de um conjunto de tabelas, cada uma designada por um nome único. O nome *Modelo Relacional* vem da correspondência entre o conceito de tabela e o conceito matemático de relação (subconjunto do produto cartesiano dos valores de uma lista de domínios), que será definido formalmente no próximo item.

Um sistema de banco de dados deve possuir a capacidade de representar dois tipos de objetos: entidades e relacionamentos. Em geral, um relacionamento é um caso especial de entidade, entretanto, o modo como os relacionamentos são representados nas abordagens dos modelos relacional, hierárquico e em redes, é distinto, enquanto que no modelo relacional são representados da mesma maneira que outras entidades, ou seja, como tuplas em relações; nos modelos hierárquicos e em redes, alguns relacionamentos podem ser representados por meio de interligações.

4.1 Relações (Tabelas)

Def.: Dada uma coleção de conjuntos $D_1, D_2, ..., D_n$, não necessariamente distintos, R é uma *relação* naqueles n conjuntos se for um conjunto de n-tuplas ordenadas $<d_1, d_2, ..., d_n>$, tais que $d_1 \in D_1$, $d_2 \in D_2$, ..., $d_n \in D_n$. Os conjuntos $D_1, D_2, ..., D_n$ são os *domínios* de R, enquanto que o valor n corresponde ao *grau* de R.

A figura 4.1 ilustra uma relação denominada *Produto*, de grau 5, onde os cinco domínios representam respectivamente, o conjunto de todos os valores de código de identificação do produto, de descrição do produto, de origem, de peso e de preço de venda. O domínio preço de venda corresponde ao conjunto de todos os valores de preços de venda possíveis para um produto, mesmo que, até o momento, existam valores de preços de venda para os quais não haja a ocorrência na relação *Produto*.

Produto_id	Descrição	Origem	Peso	Preço
A001	Produto A	Nacional	10	20,00
A002	Produto B	Nacional	5	25,00
A003	Produto C	Importado	10	10,00
B001	Produto A	Importado	5	25,00
C001	Produto B	Importado	15	15,00
C002	Produto D	Nacional	5	30,00

Fig. 4.1 – *Relação Produto*

Conforme a figura 4.1, é conveniente representar uma relação na forma de uma tabela, de modo que cada linha corresponda a uma n-tupla da relação. O número de n-tuplas (chamadas simplesmente por tuplas) de um relação é chamada de *cardinalidade* da relação. No exemplo, a cardinalidade da relação *Produto* é 6.

Relações de grau 1 são chamadas de *unárias*; relações de grau 2, são chamadas de *binárias*; de grau 3, *ternárias*; ... ; de grau n, *n-árias*.

É possível também, definir relação através do conceito de *produto cartesiano*. Dada uma coleção de conjuntos $D_1, D_2, ..., D_n$, não necessariamente distintos, o *produto cartesiano* desses n conjuntos é dado pelo conjunto de todas as combinações de valores de $<d_1, d_2, ..., d_n>$, tal que $d_1 \in D_1, d_2 \in D_2, ..., d_n \in D_n$. A figura 4.2 exibe o produto cartesiano de dois conjuntos, P e F.

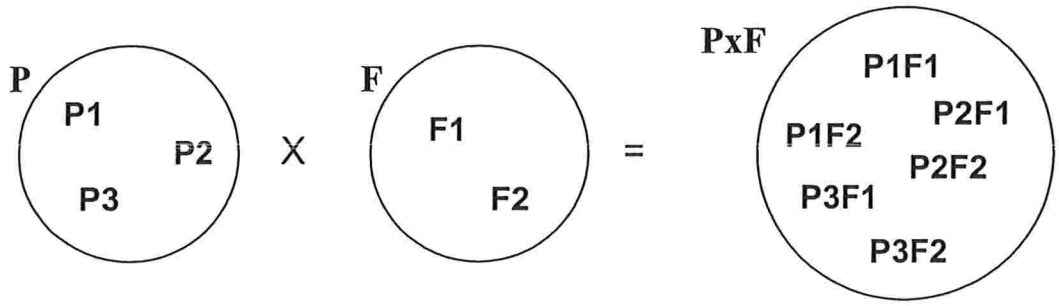


Fig. 4.2 – *Produto cartesiano PxF dos conjuntos P e F*

Agora é possível definir uma *relação R* dos conjuntos $D_1, D_2, ..., D_n$, como sendo um subconjunto do produto cartesiano de D_1 x D_2 x ... x D_n. Na figura 4.3, *PxF'* e *PxF''* representam relações dos conjuntos P e F da figura 4.2.

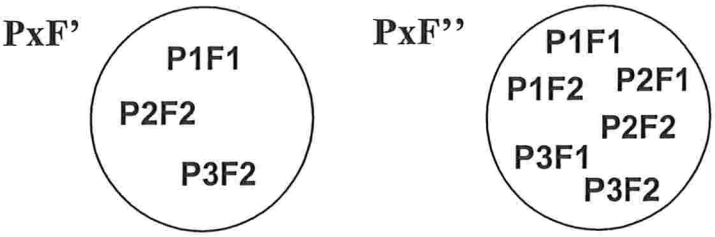


Fig. 4.3 – *Relações PxF' e PxF'' do produto*

Como uma relação é um conjunto e conjuntos não são ordenados, da mesma forma, não há ordenação entre as tuplas de uma relação, na figura 4.1, as tuplas da relação *Produto* poderiam ter sido exibidas na seqüência inversa e, ainda assim, continuaria representando a mesma relação. Entretanto, é extremamente conveniente a possibilidade de obter-se a relação segundo uma ordenação, de modo a, dada uma tupla, ter o controle sobre qual a próxima tupla, ou a tupla anterior, segundo uma ordenação crescente ou decrescente. Esta funcionalidade está presente em todos os sistemas gerenciadores de banco de dados.

Por outro lado, de acordo com a definição de relação, os domínios que a compõem respeitam uma ordenação definida entre eles, uma relação é um conjunto de n-tuplas ordenadas, sendo que o j-ésimo elemento de cada tupla pertence ao j-ésimo domínio. Se entre os n domínios $D_1, D_2, ..., D_n$, de uma relação R, existirem D_i e D_j, tal que $D_i \neq D_j$ com $i \neq j$, ao rearranjar as colunas de R de modo que D_i ocupe a j-ésima posição e D_j na i-ésima posição, a relação resultante R' seria tal que $R' \neq R$. Ou seja, se, por exemplo, as 6 colunas da tabelas *Produto* fossem rearranjadas, em uma ordenação diferente, a relação resultante seria uma relação diferente da relação *Produto*. Entretanto, a maioria dos sistemas gerenciadores de banco de dados referenciam as colunas através de um identificador único, um nome ou código, e não através de sua posição na relação. Isto permite minimizar a restrição da posição que um campo ocupa na relação, tratando a ordem das colunas de forma tão irrelevante quanto a ordem das tuplas.

Este método, utilizado por sistemas de banco de dados, de associar um nome ou código a cada domínio dos atributos das relações, foi formalizado em [Ullman 88]. Conforme foi visto, segundo a definição matemática de relação, também chamada por *set-of-lists* ou *conjunto-de-listas*, a ordem das colunas de uma relação torna-se importante. Segundo a definição proposta por Ullman, certas tabela representam a mesma informação, embora possam representar diferentes relações dentro da definição matemática de uma relação.

Por exemplo, a relação *Produto* pode ser representado, segundo a visão de *set-of-mapping* ou *conjunto-de-mapeamento*, como um *mapeamento* μ, onde a tupla (A003, Produto C, Importado, 10, 10.00) é definida por μ *(Produto_id)* = A003, μ *(Descrição)* = Produto C, μ *(Origem)* = Importado, μ *(Peso)* = 10 e μ *(Preço)* = 10.00. Neste caso, a ordem com que a tupla é listada não faz diferença, entretanto, sob a definição matemática de uma tupla, como uma lista de valores, as tuplas (A003, Produto C, Importado, 10, 10.00) e (Importado, A003, Produto C, 10, 10.00) não seriam a mesma. Assim como as relações (Produto_id, Descrição, Origem, Peso, Preço) e (Origem, Produto_id, Descrição, Peso, Preço) não seriam as mesmas.

Há situações em que a representação de uma tupla como um conjunto-de-listas é mais conveniente, como é o caso do seu uso em álgebra relacional; entretanto, existem outras situações em que é mais conveniente a utilização da representação da tupla como um conjunto-de-mapeamentos, como é o caso dos sistemas de banco de dados relacionais. Felizmente, existe uma forma óbvia de conversão entre as representações, como segue:

Dada um relação na forma conjunto-de-listas, pode-se atribuir arbitrariamente um nome a cada atributo da relação, consecutivamente a relação passa a ser representada segundo a forma de um conjunto-de-mapeamentos. Por outro lado, dada uma relação na forma de um conjunto-de-mapeamentos, pode-se fixar uma ordem arbitrária para os atributos e convertê-los para uma representação na forma de um conjunto-de-listas.

Os sistemas de banco de dados relacionais utilizam esta representação de um conjunto-de-mapeamentos, referenciando as colunas através de um identificador único, permitindo assim, minimizar a restrição da ordem dos atributos de uma relação. Sendo os sistemas de banco de dados relacionais constituídos por um conjunto de *tabelas*, que correspondem às relações, onde cada tabela é identificada por um nome único. As tabelas são constituídas por *colunas*, correspondentes aos atributos, e *linhas*, correspondentes às tuplas.

4.2 Domínios e Atributos (Colunas)

É importante observar a diferença entre o conceito de *domínio* e *atributo*, ou *coluna*, que corresponde a um valor pertencente ao *domínio*. Para enfatizar a distinção destes conceitos, na figura 4.4 foram dados nomes distintos aos atributos e aos domínios.

Domínios da relação *Produto*:

Domínio	Código_do_Produto	Texto(4)
Domínio	Descrição_do_Produto	Texto(35)
Domínio	Origem	Texto(2)
Domínio	Peso	Número(6,2)
Domínio	Preço_de_Venda	Monetário

```
Relação Produto (
        Produto_id:   Domínio    Código_do_Produto,
        Descrição:    Domínio    Descrição_do_Produto,
        Origem:       Domínio    Origem,
        Peso:         Domínio    Peso,
        Preço:        Domínio    Preço_de_Venda)
```

Fig. 4.4 – *Conceito de Domínio e Atributo*

Usualmente, a especificação explícita de atributos que assumem valores de um mesmo domínio, é omitida, atribuindo-se o mesmo nome aos atributos; no entanto nem sempre isto é possível, conforme a figura abaixo.

Loja Filial	Loja	Vendas
LJ100	LJ100	1000
LJ100	LJ101	500
LJ200	LJ200	2000
LJ300	LJ300	300
LJ300	LJ301	800
LJ300	LJ302	500
LJ400	LJ400	1000
LJ400	LJ401	600

Fig. 4.5 – *A Relação Vendas por Loja e Filial*

Na relação da figura 4.5, cada tupla é composta por três atributos: *loja filial*, *loja* e *vendas*, no entanto, a relação apresenta apenas dois domínios distintos: código do estabelecimento comercial e número de vendas, pois de acordo com a definição de relação, não há a necessidade de os domínios que constituem a relação serem distintos. A relação da figura 4.5 representa as vendas segundo a hierarquia das lojas, onde o significado de cada tupla da relação é que cada loja pertence a uma filial, sendo que uma filial é também uma loja, e para cada loja é mostrado o número de vendas realizadas.

O exemplo exibe também, a convenção comumente utilizada de adotar nomes distintos de atributos prefixando-se o nome do domínio com *nomes funcionais,* a fim de indicar as diferentes funções assumidas pelo domínio em cada uma de suas participações na relação.

4.3 Normalização

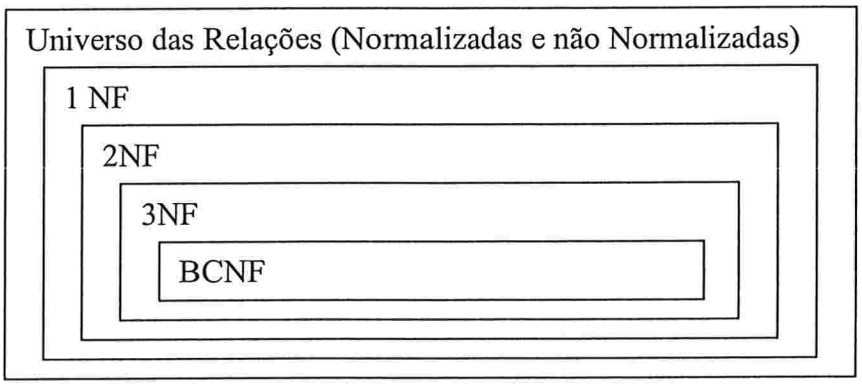


Fig. 4.6 – *Universo das Relações*

O processo de *normalização* foi inicialmente proposto por Codd em 1972, como uma série de regras aplicáveis ao esquema relacional que, quando obedecidas, resultavam em uma *forma normal*, ou seja, em um esquema bem projetado. Codd definiu três formas normais, chamadas de primeira (1NF), segunda (2NF) e terceira (3NF) forma normal. Posteriormente, Boyce e Codd propuseram uma definição mais forte da 3NF, chamada forma normal de Boyce-Codd (BCNF). Todas estas formas normais são baseadas nas dependências funcionais dos atributos das relações. A quarta forma normal (4NF) e a quinta forma normal (5NF) foram propostas depois, baseadas nos conceitos de dependências multivaloradas e dependências de junção, respectivamente. Estas últimas, devido a sua complexidade, não serão definidas a seguir, mas podem ser encontradas em [Date 86] e [Korth e Silberschatz 95].

Dizemos que uma relação *R* se encontra na *primeira forma normal* (1NF) quando todos atributos de *R* são atômicos e mono-valorados. Um domínio é dito *atômico* quando seus elementos são considerados unidades indivisíveis na base.

Portanto, uma relação é dita estar na primeira forma normal se para cada atributo das tuplas de uma relação, exista precisamente um único valor indivisível, e nunca um conjunto de valores. Por exemplo, no caso de uma relação representar um documento, onde haja o atributo autor, para que esta relação esteja na 1NF, este atributo deve possuir sempre um único nome de autor, nunca uma lista de autores.

Entretanto, são permitidos valores especiais, como é o caso do valor nulo, que representem as informações de *inaplicável* ou *desconhecido* para os valores de um atributo, como é o caso de "consumo de combustível" para o veículo bicicleta, ou "valor da comissão" para um empregado não comissionado.

É possível passar uma relação que não esteja na primeira forma normal para uma forma normal equivalente, a figura 4.7 representa a relação *Documento* definida nos domínios Título do Documento, Lista de Autores e Lista de Assuntos, onde os elementos de Lista de Autores e Lista de Assuntos são, na verdade, relações definidas nos domínios de Autores e Assuntos respectivamente, ou seja, a relação *Documento* não se encontra na 1NF. A relação *Documento'*, também presente na figura 4.7, representa uma relação equivalente a relação *Documento*, estando porém na 1NF.

Relação *Documento*:

Título	Autor	Assunto
Sistemas de Banco de Dados	{Korth, Silberschatz}	Computação, Banco de Dados
Vidas Secas	{Graciliano Ramos}	Literatura, Nordeste
The Java Programming Language	{ Ken Arnold, James Gosling}	Computação, Linguagem de Programaçao

Relação *Documento'*:

Título	Autor	Assunto
Sistemas de Banco de Dados	Korth	Computação
Sistemas de Banco de Dados	Korth	Banco de Dados
Sistemas de Banco de Dados	Silberschatz	Computação
Sistemas de Banco de Dados	Silberschatz	Banco de Dados
Vidas Secas	Graciliano Ramos	Literatura
Vidas Secas	Graciliano Ramos	Nordeste
The Java Programming Language	Ken Arnold	Computação
The Java Programming Language	Ken Arnold	Linguagem de Programação
The Java Programming Language	James Gosling	Computação
The Java Programming Language	James Gosling	Linguagem de Programação

Fig. 4.7 – *Normalização de uma relação*

Embora ambas as relações possuam grau 3, a relação *Documento* apresenta atributos cujo domínio não é *simples*, um domínio é dito *simples* quando todos seus elementos são atômicos, por outro lado, a relação *Documento'* é uma relação equivalente à relação *Documento* onde todos seus domínios são simples, ou seja, respeita a 1NF.

O objetivo de adotar a 1NF nas relações é o de simplificar a estrutura de dados para representá-las, além de simplificar as operações da linguagem de manipulação de dados.

A seguir serão definidas a 2ª e 3ª formas normais e a BCNF. Para isso, é necessário definir primeiro dependência funcional. A *dependência funcional* $X \rightarrow Y$ é válida para uma relação R se, para todos pares de tuplas t_1 e t_2 de R tal que $t_1[X] = t_2[X]$, então $t_1[Y] = t_2[Y]$.

Uma relação R está na *2NF* se, e somente se, estiver na 1NF e todos os atributos não chave forem totalmente dependentes da chave primária. Um atributo é dito *não chave* quando não participa da chave primária. Uma dependência funcional $X \rightarrow Y$ é chamada *totalmente dependente* se removendo qualquer atributo A de X, a dependência deixa de ser válida.

Uma relação *R* está na *3NF* se, e somente se, estiver na 2NF e todos os atributos não chave forem dependentes não transitivos da chave primária. Uma dependência funcional $X \rightarrow Y$ de uma relação é uma *dependência transitiva* se existe um conjunto de atributos *Z* que não é um subconjunto das chaves da relação e tal que $X \rightarrow Z$ *e* $Z \rightarrow Y$.

Uma relação *R* encontra-se na *BCNF* (forma normal de Boyce/Codd) se, e somente se, cada determinante forma uma chave candidata. Um atributo, possivelmente composto, do qual algum outro atributo é funcionalmente dependente, é chamado *determinante funcional.*

4.4 Chaves

Freqüentemente há a necessidade de se identificar as tuplas de uma relação, que pode estar representando uma entidade ou um relacionamento. Na perspectiva do banco de dados, esta identificação deve ser realizada em termos de seus atributos.

Uma *superchave* corresponde ao conjunto de um ou mais atributos que permitem identificar unicamente uma tupla de uma relação. No exemplo 4.1, o atributo Produto_id identifica unicamente um produto dos demais na relação *Produto*, portanto o conjunto de atributos {Produto_id} é uma superchave. Da mesma forma, o conjunto de atributos {Produto_id, Descrição} é outra superchave da relação *Produto*, pois identifica unicamente cada uma das tuplas da relação *Produto*, por outro lado, o conjunto de atributos {Descrição} não é uma superchave, pois diversos produtos podem possuir a mesma descrição, ou seja, mais de uma tupla da relação *Produto* pode possuir o mesmo valor para o atributo Descrição.

Portanto, dado um conjunto *K* de atributos de uma relação *R* de modo que seja uma superchave de *R*, então qualquer *superconjunto* de *K*, isto é, qualquer conjunto de atributos da relação *R* que contenha *K*, também é uma superchave. Em geral, deseja-se obter superchaves tais que não contenham nenhum *subconjunto* próprio que seja uma superchave, isto é, deseja-se o conjunto de superchaves $\{ K_1, K_2, ..., K_n \}$, tal que para cada K_i, com $i = 1, 2, ..., n$, não existe $K_i' \subset K_i$ tal que K_i' é superchave. Cada uma das superchaves mínimas K_i do conjunto $\{ K_1, K_2, ..., K_n \}$ é chamada de *chave candidata.*

Conjuntos de atributos distintos de uma relação podem ser chaves candidatas, no exemplo 4.1, o conjunto de atributos {Descrição, Origem} é suficiente para distinguir cada uma das tuplas da relação *Produto*, então tanto o conjunto {Produto_id} como {Descrição, Origem} são chaves candidatas, embora o conjunto {Produto_id, Descrição} possam identificar unicamente um tupla da relação *Produto*, este conjunto não é uma chave candidata, uma vez que o conjunto composto apenas pelo atributo Produto_id é uma chave candidata.

No projeto de um sistema de banco de dados, uma das chaves candidatas é escolhida para identificar unicamente cada tupla de uma relação, esta chave candidata é chamada de *chave primária*. As demais chaves candidatas que não são a chave primária, são chamadas *chaves alternativas*. No exemplo 4.1, sendo a chave candidata {Produto_id} escolhida para chave primária, a chave candidata {Descrição, Origem} torna-se uma chave alternativa.

Nem sempre uma relação possui uma *chave primária simples*, isto é, composta por um atributo único; muitas vezes é necessária uma combinação de atributos para definir a chave primária, neste caso, é chamada de *chave primária composta*.

Segundo a definição de relação, a existência de uma combinação de atributos de uma relação que identifique uma tupla é garantida pelo fato de uma relação ser um conjunto e, como tal, não permitir duplicidade, desta forma, sempre é possível identificar unicamente uma tupla através de todos os atributos que a compõe. Em geral, consegue-se identificar de forma única uma tupla de uma relação através de uma combinação de um número menor de atributos.

No entanto, no modelo entidade-relacionamento, nem sempre os atributos de uma entidade são suficientes para formar uma relação que possua uma chave primária. Tais conjuntos de entidades recebem o nome de *conjunto de entidades fraco*. Um conjunto de entidades que possua uma chave primária é dito *conjunto de entidades regular*. No exemplo 4.8, o conjunto de entidades *Operação* é composto pelos atributos: Código_da_Operação, Produto_id, Quantidade, Data. Embora as operações sejam distintas, operações de lojas distintas podem possuir o mesmo número, de modo que este conjunto de entidades não possui chave primária, sendo portanto um conjunto de entidades fraco. O significado de um conjunto de entidades fraco é representar um relacionamento do tipo um-para-muitos (1:n). Em especial, este tipo de relacionamento não apresenta atributos descritivos, uma vez que qualquer atributo deve estar presente no conjunto de entidades fraco. Um elemento do conjunto de entidades regular é por definição uma *entidade dominante*, enquanto que um elemento do conjunto de entidades fraco, corresponde a uma *entidade subordinada*.

Loja	Código da Operação	Produto Id	Quantidade	Data
LJ100	1	A001	5	01/02/2001
LJ100	2	C002	10	02/02/2001
LJ100	3	A001	5	02/02/2001
LJ101	1	A001	15	01/02/2001
LJ101	2	C002	10	03/02/2001

Fig. 4.8 – *Relação Operação*

Como um conjunto de entidades fraco não apresenta uma chave primária, é necessária uma forma de distinção entre as entidades do conjunto de entidades fraco que dependa de uma entidade forte em particular. O *discriminador*, corresponde ao conjunto de atributos do conjunto de entidades fraca que permita que a distinção dos elementos seja feita. No exemplo 4.8, o atributo Código_da_Operação é o discriminador do conjunto de entidades fraco OPERAÇÃO, uma vez que para cada loja, o código da operação identifica univocamente uma operação.

A chave primária de um conjunto de entidades fraco é constituído pela chave primária do conjunto de entidades regular do qual é *dependente de existência*, mais seu discriminador. No exemplo 4.8, a chave primária é dada por {Loja, Código_da_Operação}, onde o atributo

Loja identifica a entidade dominante para cada operação e o código da operação distingue, para cada loja, as operações realizadas.

A chave primária é utilizada para identificar univocamente uma entidade dentro do conjunto de entidades, similarmente, é necessário identificar univocamente um relacionamento de um conjunto de relacionamentos. Seja R um relacionamento envolvendo os conjuntos de entidades $E_1, E_2, ..., E_n$, onde *Chave primária*(E_i) designa o conjunto de atributos que formam a chave primária do conjunto de entidades E_i. Assumindo que os nomes dos atributos de todas as chaves primárias sejam únicos, caso contrário, basta utilizar um esquema apropriado para renomeá-los. Supondo que R não tem atributos, então os atributos que descrevem os relacionamentos individuais do conjunto R, representado por *atributo*(R) são:

$$\textit{Chave primária}(E_1) \cup \textit{Chave primária}(E_2) \cup ... \cup \textit{Chave primária}(E_n)$$

Caso R tenha atributos descritivos, digamos $\{ a_1, a_2, ..., a_m \}$ então o conjunto *atributo*(R)

$$\textit{Chave primária}(E_1) \cup \textit{Chave primária}(E_2) \cup ... \cup \textit{Chave primária}(E_n) \cup \{ a_1, a_2, ..., a_m \}$$

Supondo o relacionamento FornecedorProduto, que associe o conjunto de entidades *Produto*, com chave primária Produto_id, fornecidos por uma entidade do conjunto de entidades Fornecedor, com chave primária Fornecedor_id, além da data de início do fornecimento Data_Inicio, o conjunto *atributo*(FornecedorProduto) é dado por: Fornecedor_id, Produto_id e Data_Inicio.

Dado um conjunto de relacionamentos R, a chave primária de R é determinada segundo a cardinalidade das associações das entidades envolvidas no relacionamento, ou seja, segundo o tipo de relacionamento das entidades envolvidas. Se o conjunto de relacionamentos R não possui atributos que o descrevem, então o conjunto *atributo*(R) forma uma superchave.

Caso o relacionamento mapeado seja do tipo n:n, esta superchave corresponde à chave primária de R. No exemplo, caso um fornecedor possa fornecer mais de um produto e um produto possa ser fornecido por mais de um fornecedor, ou seja o relacionamento é do tipo n:n, então a chave primária é dada pelo conjunto de atributos {Fornecedor_id, Produto_id}.

Caso o relacionamento mapeado seja do tipo 1:n, a chave primária é dada pela chave primária da entidade de menor cardinalidade. No exemplo, supondo que cada produto possa ser fornecido por apenas um fornecedor, ou seja, o relacionamento é do tipo 1:n de Fornecedor para Produto, então a chave primária é dada por {Produto_id}, uma vez que cada Produto é fornecido por no máximo um Fornecedor.

Se o conjunto de relacionamentos R possuir um conjunto de atributos A associados a ele, então uma superchave é formada pelas chaves primárias das entidades envolvidas com a adição, ou não, de um ou mais atributos de A. Da mesma forma, a composição da chave primária de R dependerá da cardinalidade das entidades envolvidas. Supondo que o

relacionamento FornecedorProduto apresente o atributo Estado_do_Produto, representando o estado do Produto fornecido pelo Fornecedor: Novo ou Usado. Se um Fornecedor pode fornecer tanto produtos novos como usados, então a chave primária de FornecedorProduto é dada por {Fornecedor_id, Produto_id, Estado_do_Produto}, ou seja pelas chaves primárias das entidades envolvidas mais o atributo Estado_do_Produto. Entretanto se um Fornecedor só pode ter um tipo de produto fornecido, então o atributo Estado_do_Produto não faz parte da chave primária do relacionamento e esta é dada apenas por {Fornecedor_id, Produto_id}.

4.5 Restrições de Integridade

Restrições de Integridade são regras que visam assegurar a consistência dos dados presentes em um banco de dados. A seguir serão discutidas as principais restrições de integridade definidas para o modelo relacional.

4.5.1 Integridade de Entidade

Até então as chaves primárias foram analisadas segundo sua definição formal, que é a de prover a identificação das tuplas de uma relação. No entanto, cada tupla representa uma entidade ou relacionamento e, portanto, a chave primária é utilizada como identificador único de uma entidade/relacionamento no conjunto de entidades/relacionamentos. No exemplo 4.2, as tuplas da relação *Produto* representam produtos distintos e os valores assumidos pelo atributo Produto_id identificam cada um dos produtos, e não somente as tuplas que os representam. Desta interpretação, surge a regra de Integridade de Entidade.

Segundo a *Regra de Integridade de Entidade* nenhum atributo que compõe a chave primária pode assumir valor nulo. Isto porque todas entidades devem ser distinguíveis, ou seja devem possuir uma identificação única e, em um banco de dados relacional, as chaves primárias apresentam esta característica. Se um identificador, ou seja a chave primária, fosse nulo, significaria que existe alguma entidade ou relacionamento sem identificação única, portanto que não pode ser distinguível das demais entidades; se duas entidades não podem ser distinguíveis uma da outra, por definição são a mesma entidade. Analogamente, identificadores *parcialmente* nulos, são proibidos.

4.5.2 Integridade Referencial

De modo semelhante, surge a Segunda Regra de Integridade que atua sobre relações que fazem referência a outras relações. Por exemplo, na figura 4.9 a relação FornecedorProduto inclui referências tanto à relação Fornecedor como à relação Produto, através dos atributos Fornecedor_id e Produto_id. Espera-se portanto, que para cada valor dos atributos Fornecedor_id e Produto_id assumido nas tuplas da relação FornecedorProduto, exista uma tupla nas relações Fornecedor e Produto, do contrário, a tupla da relação FornecedorProduto estaria se referindo a um Fornecedor ou Produto inexistente.

Fornecedor Id	Produto Id
F001	A001
F001	A002
F002	A002
F001	A003
F001	B001
F002	B001
F004	B001
F003	C001
F001	C002
F002	C002

Fig. 4.9 – *Relação FornecedorProduto*

Primeiramente é necessário definir o conceito de *domínio primário*, um determinado domínio é considerado *primário* se, e somente se, existir alguma chave primária de atributo único no domínio. Por exemplo, o domínio Produto_id pode ser considerado primário, como segue

```
Domínio    Produto_id    texto(4)  Primary Key
```

Portanto, qualquer relação que inclua um atributo que esteja definido em um domínio primário deve obedecer à restrição de Integridade Referencial.

Segundo a *Regra de Integridade Referencial*, dado um domínio primário D e seja R_1 uma relação com um atributo A definido em D, então em qualquer momento, cada valor A de R_1 deve ser nulo ou igual a V, onde V é o valor da chave primária de alguma tupla de uma relação R_2, com R_1 e R_2 não necessariamente distintos, com chave primária definida em D. R_2 existe pela definição de domínio primário. Caso o atributo A seja a chave primária de R_1, a restrição é trivialmente satisfeita.

Os atributos como A, são chamados por *chave estrangeira*. Por exemplo, o atributo *loja* da figura 4.5 corresponde a uma chave estrangeira, pois os valores que este atributo pode assumir correspondem aos valores da chave primária de uma outra relação, a relação *Lojas*, que representa a entidade loja com seus atributos e características.

Através das chaves primárias e estrangeiras, representa-se os relacionamentos entre as tuplas. Na figura 4.5, o atributo *vendas* é um exemplo de atributo de relacionamento. Na figura 4.1, o atributo *origem* da relação *Produto* não é uma chave estrangeira, ele poderia tornar-se caso fosse criada uma relação *Origens*, tendo a atributo *origem* como chave primária desta relação.

4.5.3 Declaração de Chave ou Restrição de Chave

A regra de integridade de *Declaração de Chave ou Restrição de Chave* determina que se deve especificar os atributos que formam uma chave candidata para um determinado conjunto de entidades, de forma que o conjunto das inserções e atualizações válidas para o conjunto de entidades seja restrito aos que não gerem duas entidades com o mesmo valor para uma chave candidata, chamada de *Restrição de Unicidade*, além disso, a restrição de chave primária implica na restrição de não nulo, regra de Integridade de Entidade, conforme visto em 4.5.1.

4.5.4 Forma de um Relacionamento

Segundo a restrição de *Forma de um Relacionamento*, deve-se definir a cardinalidade de um relacionamento entre entidade, determinando se, no caso de relacionamentos binários, um relacionamento é do tipo muitos-para-muitos (n:n), um-para-muitos (1:n) ou um-para-um (1:1), a definição de cada um destes tipos está presente no item 3.2.

Os relacionamentos um-para-muitos e um-para-um, são restritos ao conjunto dos relacionamentos válidos entre as entidades de uma coleção de conjuntos de entidades.

4.5.5 Restrições de Domínio

Como já foi visto no item 3.3, todo atributo deve ser associado a um domínio de valores válidos para o atributo. *Restrições de domínio* são regras de integridade que atuam sobre o domínio dos atributos que compõem os conjuntos de entidades e relacionamentos. É portanto, a forma mais elementar de restrição de integridade, sendo facilmente verificada pelo banco de dados em todas as operações de inserção de dados.

4.5.6 Tipos de Domínios

É possível que atributos distintos sejam definidos sobre o mesmo domínio. Por exemplo os atributos nome do cliente e nome do funcionário podem ser definidos sobre o mesmo domínio, o conjunto de todos os nomes de pessoas. Entretanto, atributos como preço do produto e nome da loja certamente serão atributos definidos em termos de domínios distintos. Contudo, podem existir atributos onde esta distinção dos domínios seja menos clara, como é o caso dos atributos nome do cliente e nome da loja, pois no momento da implementação em um banco de dados, ambos os atributos são definidos como cadeias de caracteres, mas consultas onde se deseje todos os clientes que possuem o mesmo nome que uma loja, não fazem sentido. Conclui-se, portanto, que no nível conceitual, ao contrário do físico, nome do cliente e nome da loja devem possuir domínios bem distintos e que Restrições de Domínio não somente permitem testar os valores inseridos no banco de dados, mas também asseguram que comparações nas consultas façam sentido.

O princípio que envolve as restrições de domínio é o mesmo utilizado pelos tipos de variáveis em linguagens de programação. Linguagens podem ser *fortemente tipadas*, ou seja, possuem um maior controle e validações sobre as operações das variáveis, de acordo com o tipo a que foram definidas. Os tipos de domínio possíveis nos sistemas de banco de dados relacionais costuma ser bem restrito, os sistemas mais recentes, como os orientados a objetos, oferecem um conjunto de tipos de domínios mais flexível, podendo ser estendidos.

Bancos de dados, assim como muitas linguagens de programação, possuem alguma tolerância com operações onde os domínios sejam distintos, mas compatíveis. É o caso de um atributo definido como inteiro pequeno e outro definido como inteiro, uma vez que um inteiro pequeno é também um inteiro, operações como comparações e outras fazem sentido. Estas operações são realizadas através de uma transformação implícita de um domínio em outro, no caso a transformação de um inteiro pequeno em um inteiro. A esta transformação implícita é dado o nome de *coerção de tipo*. Outro exemplo de coerção de tipo é aplicada sobre atributos definidos em domínios de cadeias de caracteres, onde embora o comprimento possa ser diferente, são considerados domínios compatíveis.

4.5.7 Valores Vazios

Esta restrição especifica se os atributos podem ou não assumir valores vazios. É possível que certos atributos possam não ser aplicáveis a determinadas entidades, nestes casos podem assumir valores nulos, no entanto o contrário também ocorre, quando o valor nulo é impróprio para determinados atributos. Nestes casos deseja-se restringir o domínio destes atributos a fim de proibir valores vazios. A necessidade de restringir o domínio de um atributo para não permitir valores nulos é muitas vezes desejada, em particular no caso de se proibir valores nulos em atributos que compõem uma chave primária, como foi discutido na restrição de entidade, no item 4.5.1.

4.6 Mapeamento do Modelo Entidade-Relacionamento para o Modelo Relacional

A seguir será discutido o processo que leva à obtenção do modelo relacional a partir do modelo entidade-relacionamento. Este processo faz parte da engenharia direta, ou seja, do desenvolvimento de um sistema de banco de dado. Como será visto, um mesmo modelo entidade-relacionamento pode ser mapeamento para diversos modelos relacionais equivalentes, onde em geral há pelo menos uma forma preferível e outras desaconselhadas, ou por duplicar dados ou por prejudicar a performance.

De modo análogo, a formalização do mapeamento do modelo entidade-relacionamento para o relacional, pode ser utilizada no processo de engenharia reversa do modelo relacional para o entidade-relacionamento, que será discutido em maiores detalhes no capítulo 6.

O projeto lógico de um banco de dados relacional, consiste em transformar as entidades, relacionamentos, atributos, generalizações e outros elementos do modelo conceitual em elementos do modelo lógico, como tabelas e campos. Observa-se porém que um mesmo

modelo conceitual pode ser mapeado em diferentes modelos lógicos equivalentes, diferindo em pontos onde optou-se por um mapeamento em vez de outro.

De forma geral, as entidades são mapeadas para tabelas e seus atributos para os campos que compõem as tabelas, embora muitas vezes essa tradução não seja trivial.

Os relacionamentos podem ser traduzidos de três modos:

- *Tabela de Relacionamento* – quando um relacionamento é mapeado para uma tabela à parte, composta pelos identificadores das entidades associadas e pelos atributos do relacionamento.

- *Adição de Colunas nas Tabelas das Entidades* – quando um relacionamento é mapeado na forma de um ou mais campos que são adicionados a uma entidade, é empregado quando a cardinalidade máxima de uma das entidades associada é no máximo 1, neste caso o identificador da entidade de cardinalidade máxima 1 e os atributos do relacionamento são adicionadas às outras entidades associadas.

- *Fusão de Entidades* – quando as entidades associadas são mapeadas para uma única tabela, contendo todos os atributos das entidades associadas e os atributos do relacionamento, este tipo de mapeamento ocorre em relacionamentos de cardinalidade 1:1

Em geral relacionamentos são binários, a tabela a seguir associa o tipo de relacionamento com a regra de implementação [Heuser 99]:

		Regra de Implementação		
Tipo de Relacionamento	**Cardinalidade**	**Tabela de Relacionamento**	**Adição de Coluna**	**Fusão de Tabela**
1:1	(0,1) ◇ (0,1)	±	√	×
	(0,1) ◇ (1,1)	×	±	√
	(1,1) ◇ (1,1)	×	±	√
1:n	(0,1) ◇ (0,n)	±	√	×
	(0,1) ◇ (1,n)	±	√	×
	(1,1) ◇ (0,n)	×	√	×
	(1,1) ◇ (1,n)	×	√	×
n:n	(0,n) ◇ (0,n)	√	×	×
	(0,n) ◇ (1,n)	√	×	×
	(1,n) ◇ (1,n)	√	×	×

Onde :

√ - Forma Preferencial

± - Forma Alternativa

× - Forma Desaconselhada

Fig. 4.10 – *Mapeamento de Relacionamentos Binários do Modelo Entidade-Relacionamento para o Modelo Relacional*

Capítulo 5

Indicadores de Qualidade

"Indicador de Qualidade é um meio de verificação do grau de qualidade de um produto ou processo" [SMA 94].

Neste capítulo, será apresentado um breve histórico da evolução dos conceitos de qualidade, inicialmente associados ao controle estatístico de produtos manufaturados, até seus conceitos mais recentes. Serão apresentados ainda, alguns de seus elementos e características, em especial será dado um enfoque aos indicadores de qualidade.

O objetivo deste capítulo é fornecer a teoria de qualidade necessária à definição formal de indicadores de qualidade aplicáveis às implementações de bancos de dados relacionais, empregando também, conceitos já expostos sobre desenvolvimento de um banco de dados relacional, presente no capítulo 2, sobre o modelo entidade-relacionamento, capítulo 3, e o modelo relacional, capítulo 4.

Os indicadores de qualidade aplicáveis a bancos de dados relacionais definidos neste capítulo serão empregados no capítulo 7, na elaboração de uma ferramenta de auxílio a obtenção dos valores destes indicadores e, no capítulo 8, onde os indicadores serão aplicados em alguns casos de uso.

O termo qualidade, embora freqüente nos dias atuais, é muitas vezes utilizado sem uma preocupação maior com o seu emprego. Visando defini-lo de forma correta e formal, neste capítulo são apresentados alguns conceitos fundamentais necessários. Maiores detalhes podem ser encontrados em [Rolt 98], [Crosby 94], [Feignbaum 86], [Juran 92], [Deming 90] e [Ishikawa 86].

5.1 Histórico da Qualidade

O termo qualidade foi inicialmente empregado nas linhas de produção das indústrias, durante as décadas de 50 e 60, onde eram realizados controles estatísticos através de inspeções dos produtos manufaturados. Este processo, chamado *Controle Estatístico de Qualidade*, ainda está presente em muitas indústrias dos dias atuais.

Com a década de 70, os conceitos de qualidade sofreram profundas transformações, sendo estendidos a outras áreas, e não somente à produção. A partir de então, os conceitos de qualidade adquiriram um enfoque de movimento de garantia da qualidade, com o objetivo da contínua adequação aos anseios dos clientes, funcionários, empresas e sociedade.

As transformações sofridas foram conseqüência do aumento da competitividade das empresas, do fenômeno da globalização da economia e da necessidade de aumentar a lucratividade através da diminuição dos custos de produção. Neste contexto, o Japão foi o

principal responsável, pois, após a Segunda Guerra Mundial, passou a investir na melhoria da qualidade de seus produtos visando a satisfação de seus clientes, principalmente o mercado dos EUA, que acabou tomando medidas de sanções às importações japonesas, e passou a incentivar investimentos na qualidade de seus produtos.

O papel da qualidade, até então enfocado em estatísticas dos processos industriais, ampliou sua abordagem para um processo de melhoria contínua da qualidade, buscando solucionar problemas envolvendo outras áreas das organizações que passaram a empregar o uso da qualidade com o objetivo de melhorar a produtividade, a qualidade dos produtos e a lucratividade.

5.2 Conceitos de Qualidade

Diversos conceitos de *qualidade* foram formulados [Rolt 98]:

- Conformidade com os requisitos [Crosby 94]
- Conjunto de características incorporadas ao produto através de projeto e manufatura que determina o grau de satisfação dos clientes [Feignbaum 86]
- Qualidade é a adequação ao uso [Juran 92]
- Perseguição às necessidades dos clientes e homogeneidade dos resultados do processo [Deming 90]
- Rápida percepção e satisfação das necessidades do mercado, adequação ao uso dos produtos e homogeneidade dos resultados do processo [Ishikawa 86]

Observa-se portanto que o conceito de qualidade é bastante amplo e muitas vezes é definido segundo um enfoque, mas em geral, qualidade é uma característica que envolve produto, processo e o atendimento das necessidades dos clientes.

Garvin resumiu os conceitos anteriores segundo cinco abordagens [Garvin 92]:

1. *Transcendente*: a qualidade é excelência nata, não sendo possível definir qualidade com precisão devido à sua simplicidade não analisável e reconhecida através de experiência.
2. *Baseada no Produto*: a qualidade se encontra presente no produto, sendo uma variável precisa e mensurável. A qualidade é vista como uma característica inerente ao produto e não como algo atribuído a ele.
3. *Baseada no Usuário*: considera a qualidade sob o ponto de vista do cliente, melhor qualidade corresponde ao pleno atendimento de suas necessidades e preferências.
4. *Baseada na Produção*: define qualidade como conformidade com as especificações. O ato de fazer certo da primeira vez é a idéia de excelência, qualquer desvio implica em perda de qualidade.
5. *Baseada no Valor*: define qualidade em relação ao custo e preço. Um produto possui qualidade se apresentar um alto grau de conformidade a um custo aceitável.

Neste trabalho, o objetivo é desenvolver uma técnica que permita comparar a qualidade da implementação de um produto, no caso um banco de dados relacional, ou seja a qualidade com que o produto foi concebido. Esta medida de qualidade possui impacto no processo de manutenção e consecutivamente, nos custos agregados; portanto segundo os conceitos resumidos de Garvin, adotamos uma abordagem baseada tanto no produto, como na produção, no valor e no usuário ou cliente, considerando o papel dos programadores e analistas que darão suporte às manutenções e alterações do sistema.

O processo de *avaliação da qualidade* de um produto, processo ou organização, é dado pela comparação e análise de *indicadores de qualidade*, que são utilizados para mensurar o grau de qualidade. Os indicadores de qualidade devem atender a certos requisitos, como simplicidade, representatividade, rastreabilidade e baixo custo dentre outros e são classificados segundo sua unidade de medida, como proporção entre um número de ocorrências perfeitas, relação entre um quantitativo e um referencial apropriado ou ainda um número absoluto ou percentual de ocorrências dentro de um período de tempo.

Indicadores de Qualidade são elementos que medem os níveis de *eficiência*, que corresponde ao modo correto de atuar ou produzir efetivamente; e *eficácia*, que corresponde à capacidade de produzir um resultado desejado, de uma organização, de um processo ou produto, sendo também designado por *Medidas de Desempenho* ou *Taxas de Melhorias*.

Em termos de qualidade, *medição* corresponde ao sistema de apoio ao planejamento, tomada de decisão e controle da qualidade [Oliveira, Latelme e Formoso 95]. A coleta, processamento e avaliação dos dados, ou seja a medição, é utilizada pelos indicadores para comparar o desempenho de uma medida em relação a uma meta estabelecida.

Os indicadores de qualidade devem possuir algumas características:

- Precisão, ou seja, não podem ser ambíguos
- Facilidade de levantamento
- Facilidade de compreensão
- Facilidade de comparação

Segundo a definição apresentada por Gil [Gil 92], os indicadores de qualidade são descritos em termos de três conceitos:

Elemento - assunto/situação base para a caracterização do indicador de qualidade. Por exemplo: peças produzidas, profissionais alocados, taxa de entidades normalizadas, taxa de sinônimos e homônimos etc.

Fator – combinação de elementos. Por exemplo: peças produzidas por máquina, profissionais alocados por área empresarial, número de entidades normalizadas sobre o número total de entidades etc.

Métrica - unidade/forma de mensuração de elementos e fatores, por exemplo: valor, quantidade, tempo, porcentagem.

Um *índice* corresponde a uma informação extraída de um conjunto de indicadores de modo a simplificar a comunicação e representação dos resultados.

Segundo [Gouzee, Mazijn e Billharz 95], os indicadores diferem de dados na forma bruta, embora muitas vezes sejam apresentados na forma estatística ou gráfica. Indicadores e índices podem ser vistos como a forma mais alta de informação, estando presente no topo da pirâmide de informação, figura 5.1 , cuja base é formada pelas informações na forma de dados brutos.

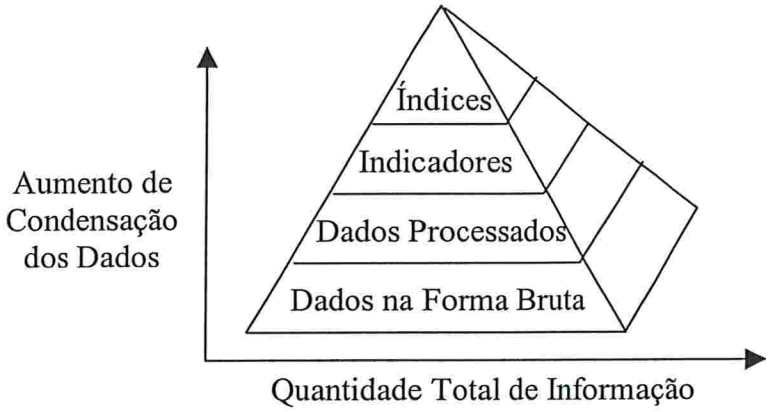


Fig. 5.1 – *Pirâmide de Informação segundo Gouzee*

Os indicadores podem ser apresentados de duas formas:

- *Indicadores Simples* – expressam um valor absoluto, por exemplo número de tabelas de um sistema, número de erros de um sistema, etc.
- *Indicadores de Razão* – expressam a relação de duas ou mais grandezas correlacionadas, por exemplo número de linhas de código / recursos alocados, número de relações normalizadas / número total de relações, etc.

E podem ter duas naturezas distintas:

- *Qualitativa* – determina o valor referente a uma graduação, por exemplo a satisfação como: 5 – Muito Boa, 4 – Boa, 3 – Aceitável, 2 – Ruim, 1 – Muito Ruim.
- *Quantitativo* – determina um valor numérico, como por exemplo tempo médio de atendimento, etc.

5.3 Definição de Indicadores de Qualidade para Implementação de Banco de Dados Relacional

Indicadores de qualidade são utilizados para mensurar a qualidade de um produto ou processo em relação a uma meta e são utilizados para comparar e acompanhar um processo ao longo do tempo.

Neste estudo, é avaliada a qualidade do produto, onde o objetivo é comparar diferentes implementações. Contudo, é possível utilizar os indicadores para avaliar e acompanhar a qualidade da implementação no decorrer do tempo, onde são realizadas alterações e adições.

Ao avaliar a qualidade da implementação de um banco de dados, alguns aspectos ou regras são analisados a fim de se quantificar e comparar a qualidade com que o software foi desenvolvido. A partir destes aspectos e regras, serão definidos os indicadores de qualidade que serão utilizados na avaliação da qualidade de uma implementação de banco de dados relacional.

No projeto de um banco de dados relacional, como cada fase do projeto resulta em um modelo, o produto da fase do projeto, é possível definir um conjunto de indicadores para cada nível do projeto de desenvolvimento do banco de dados:

- *Indicadores de Qualidade do Nível Conceitual* – são indicadores que avaliam aspectos pertinentes ao Modelo Conceitual de um Projeto de Banco de Dados. Por exemplo: indicadores que avaliem Normalização, Redundância ou Perda de Informação, etc.

- *Indicadores de Qualidade do Nível Lógico* – correspondem a indicadores que são utilizados para avaliar aspectos da qualidade do Modelo Lógico de um Projeto de Banco de Dados. Exemplo: Presença de Sinônimos, Presença de Homônimos, Diferença de Domínio, Segurança e Rastreabilidade, etc.

- *Indicadores de Qualidade do Nível Físico* – indicadores que são utilizados para avaliar aspectos do Modelo Físico de um Projeto de Banco de Dados. Exemplo: Número de Índices, Índices Desnecessários, etc.

Abaixo são propostos alguns indicadores para avaliar a qualidade do modelo conceitual entidade-relacionamento de um banco de dados relacional:

Elemento	Fator	Medida
Taxa de Infração de 1NF	nº de Relações que não respeitam a 1NF / nº Total de Relações	%
Taxa de Infração de 2NF	nº de Relações que não respeitam a 2NF / nº Total de Relações	%
Taxa de Infração de 3NF	nº de Relações que não respeitam a 3NF / nº Total de Relações	%
Taxa de Infração de BCNF	nº de Relações que não respeitam a BCNF / nº Total de Relações	%
Taxa de Infração de 4NF	nº de Relações que não respeitam a 4NF / nº Total de Relações	%
Taxa de Infração de 5NF	nº de Relações que não respeitam a 5NF / nº Total de Relações	%

Fig. 5.2 – *Tabela de Indicadores de Qualidade do Nível Conceitual de um Projeto de Banco de Dados*

Abaixo são propostos alguns indicadores do nível Lógico que podem ser definidos a partir do mapeamento dos relacionamentos binários do modelo entidade-relacionamento para o modelo relacional proposto por [Heuser 99] e presente no capítulo 4.

Elemento	Fator	Medida
Taxa de Relacionamentos de Cardinalidade(X) Mapeados na Forma(Y)	nº de Relacionamentos de Cardinalidade(X) Mapeados na Forma(Y) / nº Total de Relacionamentos de Cardinalidade(X)	%

Onde,

X = { {(0,1),(0,1)}, {(0,1),(1,1)}, {(1,1),(1,1)},
{(0,1),(0,n)}, {(0,1),(1,n)}, {(1,1),(0,n)}, {(1,1),(1,n)},
{(0,n),(0,n)}, {(0,n),(1,n)}, {(1,n),(1,n)} }, e

Y = { Preferencial, Alternativa, Desaconselhada }

Fig. 5.3 – *Tabela de Indicadores de Qualidade do Mapeamento de um Relacionamento*

No entanto, a análise deste conjunto de indicadores necessita de um conhecimento do modelo Conceitual, que assumimos neste estudo não existir. Por isto, estes indicadores são apenas citados, embora sejam de grande utilidade, pois exibem problemas no projeto do sistema, podendo ocorrer tanto perda de informação como em outros casos, desperdício de espaço e perda de performance do sistema.

A seguir são definidos outros indicadores para avaliar a qualidade do nível lógico do projeto de um banco de dados relacional. Estes indicadores serão detalhados e estudados nos itens seguintes deste capítulo:

Elemento	Fator	Medida
Taxa de Homônimos	nº de Homônimos / nº Total de Nomes	%
Taxa de Sinônimos	nº de Sinônimos / nº Total de Nomes	%
Taxa de Divergência de Domínio	nº Domínios Divergentes / nº Total de Domínios	%
Taxa de Ausência de Informações de Segurança e Rastreabilidade	nº de Ausência de Informações de Segurança e Rastreabilidade nas Tabelas / (nº Total de Tabelas * nº de Informações de Segurança e Rastreabilidade)	%

Fig. 5.4 – *Tabela de Indicadores de Qualidade do Nível Lógico de um Projeto de Banco de Dados*

O indicador Taxa de Ausência de Informações de Segurança e Rastreabilidade pode ser aplicado a vários itens que podem estar presentes ou ausentes nos sistemas, tais como: usuário que realizou o cadastro da informação, data de cadastro, usuário que realizou a última alteração, data da última alteração, identificação da aplicação que realizou a última alteração. Portanto, o valor deste indicador pode ser definido através da composição dos valores dos seguintes itens:

Elemento	Fator	Medida
Taxa de Ausência de Identificação de Usuário Cadastrante	nº Ausências de Identificação do Usuário Cadastrante / nº Total de Tabelas	%
Taxa de Ausência da Data de Cadastro	nº Ausências de Data de Cadastro / nº Total de Tabelas	%
Taxa de Ausência de Identificação de Usuário da Última Alteração	nº Ausências de Identificação de Usuário da Última Alteração / nº Total de Tabelas	%
Taxa de Ausência da Data da Última Alteração	nº Ausência de Data de Última Alteração / nº Total de Tabelas	%
Taxa de Ausência de Identificação da Aplicação da Última Alteração	nº Ausência de Identificação da Aplicação de Última Alteração / nº Total de Tabelas	%

Fig. 5.5 – *Tabela de Indicadores de Qualidade do Nível Lógico de um Projeto de Banco de Dados*

Portanto, pode-se definir o indicador Taxa de Ausência de Informações de Segurança e Rastreabilidade como sendo:

Elemento	Fator	Medida
Taxa de Ausência de Informações de Segurança e Rastreabilidade	$\dfrac{\sum_{i=1}^{n} \text{nº Ausências do Campo}(x_i)}{\sum_{i=1}^{n} \text{nº Tabelas do Sistema}}$	%

Fig. 5.6 – *Taxa de Ausência de Informações de Segurança e Rastreabilidade*

Devido a simplicidade de entendimento da definição de alguns indicadores, como é o caso dos indicadores do modelo conceitual entidade relacionamento e dos indicadores do

mapeamento do modelo entidade-relacionamento para o relacional, não serão definidos a seguir, sendo apenas os que apresentarem conceitos ou elementos ainda não definidos em capítulos anteriores. Detalhes sobre as formas normais, podem ser encontrados no item 4.3, e sobre o mapeamento do modelo entidade-relacionamento para o modelo relacional, no item 4.6.

5.3.1 Taxa de Homônimos e Sinônimos

Homônimos – objetos que embora distintos, apresentam o mesmo nome.

Sinônimos – nomes distintos que são empregados para referenciar o mesmo objeto.

A presença de homônimos e sinônimos é uma característica indesejada nos sistemas, mas apesar da simplicidade e facilidade em evitá-las, ocorrem com muita freqüência em sistemas reais. Embora a presença desta característica não impacte na performance do sistema, dificulta a compreensão e aumenta o custo de manutenção dos sistemas, pois torna mais difícil o entendimento do sistema facilitando a ocorrência de erros. A meta para estes indicadores é tender a 0%.

5.3.2 Taxa de Divergência de Domínios

Esta característica é ocasionada quando são empregados domínios diferentes para representar os valores de um mesmo atributo em entidades distintas, ou seja, o mesmo atributo é representado por tipos/tamanhos de dados diferentes em entidades distintas.

Esta característica, a exemplo da anterior, pode ser facilmente evitada, mas ocorre com freqüência nos sistemas reais, principalmente em sistemas legados. Com isto aumenta-se a dificuldade de entendimento, pois gera-se a dúvida se os atributos representam o mesmo significado ou não, além de gerar um tratamento especial desnecessário para a conversão de tipos e verificação de limites de validade, aumentando a possibilidade de erros e o número de operação, prejudicando a performance do sistema. Novamente, a meta para este indicador é tender a 0%.

5.3.3 Taxa de Ausência de Informações de Segurança e Rastreabilidade

Este indicador é utilizado para analisar a facilidade de identificação de aplicativos geradores de erros e inconsistências, além de usuários que alteraram informações ou fraudaram o sistema.

Com o barateamento e popularização das redes e desenvolvimento de sistemas em várias camadas, os sistemas tornaram-se multi-usuários e interconectados, trazendo a segurança da informação a uma posição de maior destaque.

É interessante que os sistemas procurem armazenar também informações sobre eles próprios, tais como: aplicativo que efetuou a operação, data, hora e usuário, em sistemas de mais de uma camada; o serviço e fonte originária de acesso, além de histórico de alterações e informações antigas.

Estas informações armazenadas que se referem às próprias informações, sua manipulação, regras de utilização ou outras características da representação e não propriamente da informação representada, são chamadas de meta-informações.

Com a presença de meta-informações torna-se mais fácil identificar, localizar e corrigir erros (é comum sistemas com mais de 100 aplicativos acessando à mesma informação, onde sem este recurso é praticamente impossível identificar qual o software propagador de erros), além de poderem ser utilizadas também em auditorias de combate a fraudes contra as corporações. A meta deste indicador é 0%.

A seguir no próximo capítulo, serão apresentadas as principais técnicas de engenharia reversa aplicáveis à modelos relacionais de banco de dados. Estas técnicas serão úteis na extração de informações necessárias para obtenção dos valores de alguns dos indicadores de qualidade de banco de dados relacionais apresentados neste capítulo. A utilização destes indicadores será descrita no capítulo 7 e aplicada em casos de uso no capítulo 8.

Capítulo 6
Técnicas de Engenharia Reversa de um Banco de Dados

"Engenharia reversa pode ajudar extraindo uma especificação conceitual, independente da implementação" [Andersson 94].

O objetivo deste capítulo é apresentar de forma simples algumas das técnicas mais empregadas na realização de engenharia reversa de banco de dados. Estas técnicas serão úteis no próximo capítulo, onde serão empregadas na implementação de uma ferramenta de auxílio a obtenção dos valores dos indicadores de qualidade de implementação de banco de dados que foram definidos no capítulo 5.

A engenharia reversa de banco de dados relacional é adotada quando se deseja obter um modelo entidade-relacionamento ou um modelo orientado a objeto. As principais técnicas de engenharia reversa empregadas em banco de dados relacionais, podem ser divididas em:

6.1 Análise de Nomes

A partir dos nomes das tabelas e campos, busca-se identificar sinônimos e homônimos para determinar entidades, atributos, relacionamento e chaves.

O objetivo desta técnica de engenharia reversa é, através da enumeração das ocorrências de nomes no esquema do sistema, produzir um dicionário de dados. Através do dicionário de dados, podem ser identificados todos os homônimos e sinônimos do sistema. Para isto devem ser avaliados os nomes e suas características, como o domínio (tipo e tamanho de dado).

Neste caso assume-se que todas as ocorrências do mesmo nome, com mesmo domínio, referem-se à mesma entidade ou informação. Caso pelo menos uma das características de tamanho ou tipo não seja coincidente, assume-se que o nome se refere a entidades ou informações distintas.

Nesta análise, é necessária a intervenção humana nos seguintes casos:

- *Sinônimos triviais* – quando o mesmo nome, embora não possuam o mesmo tipo ou tamanho, refere-se à mesma entidade ou informação.
- *Sinônimos* – ou seja, nomes distintos que se referem à mesma entidade ou informação.
- *Homônimos* – quando o mesmo nome se refere a entidades ou informações distintas.

O fato de existirem muitas ocorrências destes casos indica uma ausência da preocupação com a nomenclatura da representação no banco de dados e conseqüente falta de qualidade do sistema produzido.

A existência deste dicionário de dados facilita inferir as chaves primárias e estrangeiras das tabelas do banco de dados do sistema.

6.2 Análise de Esquemas

Esta técnica visa obter o modelo entidade-relacionamento de um banco de dados a partir de sua estrutura, ou seja de seu esquema, podendo ser dividida nos seguintes passos:

- Identificação das entidades e atributos na construção de um modelo entidade-relacionamento correspondente a cada tabela;
- Definição dos relacionamentos 1:n e 1:1;
- Definição de atributos de relacionamentos;
- Definição de identificadores de entidades e relacionamentos.

6.2.1 Identificação e Construção de Modelo ER Correspondente a Cada Tabela

Uma tabela pode corresponder a:

- Uma entidade;
- Um relacionamento;
- Uma entidade especializada.

As tabelas podem ser classificadas pelos seguintes tipos:

Relacionamento – correspondem às tabelas cuja chave primária é composta por várias chaves estrangeiras. É utilizada para implementar um relacionamento n:n entre as entidades cujas chaves primárias constam como chaves estrangeiras da chave primária da tabela de relacionamento.

Entidade Especializada – corresponde à tabela cuja chave primária é também uma chave estrangeira, ou seja, corresponde à chave primária de outra tabela. Caracteriza a especialização da entidade representada pela tabela referenciada pela chave estrangeira.

Entidades – nos demais casos, as tabelas representam uma entidade.

6.2.2 Definição dos Relacionamentos 1:n e 1:1

Tabelas que possuam chaves primárias que sejam chave estrangeira e não façam parte da chave primária representam relacionamentos 1:1 e 1:n. Através da análise dos dados contidos nestes campos pode-se inferir a cardinalidade do relacionamento.

6.2.3 Definição dos Atributos de Relacionamentos

Os campos das tabelas de relacionamento que não fazem parte da chave primária, representam atributos do relacionamento.

6.2.4 Definição de Identificadores de Entidades e Relacionamentos

As colunas das chaves primárias das tabelas que não são chave estrangeira, caracterizam atributos identificadores da entidade ou relacionamento.

As colunas da chave primária das tabelas que são chaves estrangeiras são identificadores externos de entidades.

6.3 Análise de Dados

A análise de dados consiste em, a partir da análise dos valores dos atributos das relações do modelo relacional armazenados no sistema, identificar atributos que se referem à mesma entidade ou informação representada. Assim como a análise de nomes, serve para identificar chaves primárias e estrangeiras, mas certamente é um método muito mais custoso em termos de processamento e análise, pois utiliza a comparação de ocorrência de valores armazenados no sistema.

Esta técnica de engenharia reversa é muito utilizada na realização da *Data Mining*, onde o interesse está em padronizar a representação de informações no sistema, facilitando a sua obtenção para, por exemplo, a construção de um DataWarehouse.

6.4 Análise das Principais Consultas do Sistema

Esta técnica consiste em definir as chaves primárias, candidatas e estrangeiras de um modelo relacional de banco de dados, a partir da análise das principais consultas e índices do sistema.

Consultas onde se busca obter uma única relação de uma entidade ou índices únicos, representam um conjunto de atributos que definem uma chave candidata a chave primária.

Consultas que realizam "joins" entre tabelas, indicam as chaves primárias destas tabelas, tabelas de relacionamento e chaves estrangeiras de tabelas.

Um modelo entidade-relacionamento, que como foi visto pertence ao Nível Conceitual de um projeto de banco de dados, pode resultar em vários modelos relacionais, pertencentes ao Nível Lógico, ou seja um mesmo modelo entidade-relacionamento pode ser implementado de formas diferentes, sendo observados os seguintes pontos:

- Todas as implementações são corretas;
- Podem resultar em performances diferentes;
- Podem resultar em uma maior/menor dificuldade de desenvolvimento/manutenção.

Neste capítulo, foram expostas de maneira superficial as principais técnicas adotadas quando se deseja realizar o processo de engenharia reversa de uma implementação de banco de dados relacional, partindo da implementação do modelo lógico visando a obtenção do modelo conceitual. Muitas das informações necessárias para a obtenção dos valores dos indicadores de qualidade propostos no capítulo 5, utilizam informações ou dados que devem ser extraídos da implementação do modelo relacional e, portanto, podem ser empregadas algumas das técnicas descritas neste capítulo. Um exemplo de emprego de técnicas de engenharia reversa é a utilização da análise de nomes para a definição da taxa de homônimos e sinônimos, conforme será tratado no próximo capítulo.

Capítulo 7
Implementação de Ferramenta de Auxílio à Análise de Qualidade de uma Implementação de Banco de Dados Relacional

"A engenharia reversa de sistemas representa um problema de grande complexidade computacional, no qual é constante a necessidade de interagir com humanos para complementar lacunas do conhecimento embutido em softwares legados. Por outro lado, é imprescindível o auxílio de ferramentas computacionais" [Freitas e Leite 97].

Neste capítulo serão definidos alguns conceitos que servirão de base para a extração de informações que serão utilizadas no cálculo dos valores dos indicadores propostos no capítulo 5. Será também, descrita a implementação de uma ferramenta de auxílio à análise da qualidade de uma implementação de banco de dados relacional, cujo objetivo é permitir que sejam mensurados alguns dos indicadores de qualidade do nível lógico, conforme foram descritos no capítulo 5, utilizando-se das técnicas de engenharia reversa descritas no capítulo 6.

7.1 Objetivos

O objetivo dessa ferramenta é mensurar alguns indicadores de qualidade a partir de uma implementação de banco de dados relacional, ou seja, a partir de tabelas, campos e tipos de dados destes campos. Os indicadores utilizados são:

a) Taxa de Homônimos;
b) Taxa de Sinônimos;
c) Taxa de Divergência de Domínio;
d) Taxa de Ausência de Informações de Segurança e Rastreabilidade.

Os itens a e b são os que apresentam a maior necessidade de intervenção humana, embora os itens c e d utilizem de forma indireta estas informações. É necessário agrupar campos que possuam o mesmo significado semântico, que embora possuam nomes, tipos, tamanhos etc. distintos representam a mesma informação, ou seja, que sejam sinônimos. Por outro lado, é necessário desagrupar os campos que embora possuam o mesmo nome, tipo, tamanho, etc., representem informações distintas, ou seja, campos que sejam homônimos.

A partir destes agrupamentos, é possível obter as informações básicas para o cálculo dos indicadores definidos no capítulo 5, tais como: número de tabelas, número de sinônimos, número de homônimos, número de campos com domínio divergente, campos que contêm informações de segurança e rastreabilidade, etc.

7.2 Conceito de Apelidos

Para realizar este agrupamento, foi definido o conceito de *aliases*, ou *apelidos*, onde cada apelido corresponde a um identificador único para uma informação. Esta informação pode estar associada a diferentes nomes de campos do banco de dados. Por exemplo, a informação "unidade federal", pode ser representada no banco de dados através de diversos nomes de campos como: unidade_federal, uf, estado_federal, estado, etc. Neste caso, a cada um destes campos será associado o mesmo apelido:

Tabela	Campo	Apelido
Tabela1	Uf	unidade_federal
Tabela2	Estado	unidade_federal
Tabela3	Estado	unidade_federal
Tabela4	estado_federal	unidade_federal

Fig. 7.1 – *Tabela de apelidos I*

Por outro lado, um mesmo nome de campo pode possuir significados completamente distintos, ou seja, um mesmo nome de campo pode possuir mais de um apelido associado. Por exemplo, o campo estado pode, em determinadas tabelas, representar a informação unidade_federal; em outras tabelas pode representar o estado físico de um item de venda, estado_do_item: novo, usado; em outras tabelas pode representar o estado de um pedido de venda, estado_do_pedido: cadastrado, aprovado, faturado, entregue, cancelado. Neste caso, cada um destes campos, embora possua o mesmo nome, estará associado a apelidos distintos:

Tabela	Campo	Apelido
Cidades	estado	unidade_federal
Itens_Estoque	estado	estado_do_item
Produtos	estado	estado_do_item
Pedidos	estado	estado_do_pedido

Fig. 7.2 – *Tabela de apelidos II*

Na verdade, o apelido serve apenas para agrupar campos que possuam o mesmo domínio e significado. Optou-se por empregar o termo apelido ao invés de domínio por dois motivos: facilitar o entendimento, evitando confusão com o termo domínio referente ao domínio original do campo analisado; e por apelidos poderem ser distintos devido aos campos possuírem significados diferentes, mesmo quando definidos sobre o mesmo domínio, como ocorre por exemplo, quando o domínio Texto de tamanho 30 é utilizado para definir o nome de pessoas que pode ser utilizado tanto na entidade cliente como na entidade usuário, no entanto o significado pertinente a cada um deles é bem distinto.

7.3 Conceito de Sinônimo e Homônimo Adotando Apelidos

Estando todos os campos classificados com os apelidos corretos, é possível identificar homônimos e sinônimos da seguinte forma:

- *Sinônimos* - são os campos que embora possuam nomes distintos, apresentam o mesmo apelido;
- *Homônimos* - são os campos que embora possuam o mesmo nome, apresentam apelidos distintos.

Nos exemplos acima, teríamos:
3 sinônimos: uf, estado e estado_federal.
3 homônimos: unidade_federal, estado_do_item e estado_do_pedido.

7.4 Conceito de Diferença de Domínio Adotando Apelidos

Os apelidos são também utilizados para identificar os diferentes tipos de dados utilizados para representar uma mesma informação, ou seja, as diferenças de domínio dos campos das tabelas. Por exemplo, a informação "unidade federal", pode ser representada no banco de dados através de campos com tamanhos diferentes: texto de tamanho 2, texto de tamanho 12, texto de tamanho 8, etc.

Tabela	Campo	Tipo de Dado	Tamanho	Apelido
Tabela1	uf	Texto	2	unidade_federal
Tabela2	estado	Texto	2	unidade_federal
Tabela3	estado	Texto	12	unidade_federal
Tabela4	estado	Texto	8	unidade_federal

Fig. 7.3 – *Tabela de Diferença de Domínio*

7.5 Segurança e Rastreabilidade Adotando Apelidos

Informações armazenadas no sistema que se referem às próprias informações, sua manipulação, regras de utilização ou outras características da representação de uma informação, como é o caso das informações de segurança e rastreabilidade, são chamadas de *meta-informações*.

As meta-informações referentes à segurança e rastreabilidade também são determinadas analisando-se os apelidos. Como espera-se que a ocorrência destas informações esteja relacionada ao número de tabelas que constituem o sistema, tendem a ser as informações de maior incidência na base de dados, ocorrendo uma vez para cada tabela. É possível então, enumerar as incidências e ausências dos apelidos por tabela, desta forma podemos identificar a presença ou não das meta-informações em todas as tabelas.

7.6 Tecnologia Adotada

A seguir serão descritos detalhes técnicos de implementação da ferramenta utilizada:

7.6.1 Arquitetura

Para implementação desta ferramenta, foi utilizada a linguagem de programação java, que proporciona uma aplicação com independência de plataforma, podendo ser executada em qualquer máquina que possua uma JVM (Java Virtual Machine) sem necessidade da aplicação ser recompilada, ou que existam versões diferentes da aplicação para cada tipo de máquina. A opção pela linguagem java foi baseada também no fato de ser uma linguagem orientada a objetos e possui uma grande quantidade de objetos já implementados, principalmente os de interface gráfica.

O acesso a base de dados é realizado por meio de drivers JDBC, o que determina uma independência do gerenciador de banco de dados, podendo ser tanto Access como Oracle, Sybase ou Informix entre outros, bastando que exista apenas uma classe java que implemente um driver JDBC para o gerenciador de banco de dados desejado.

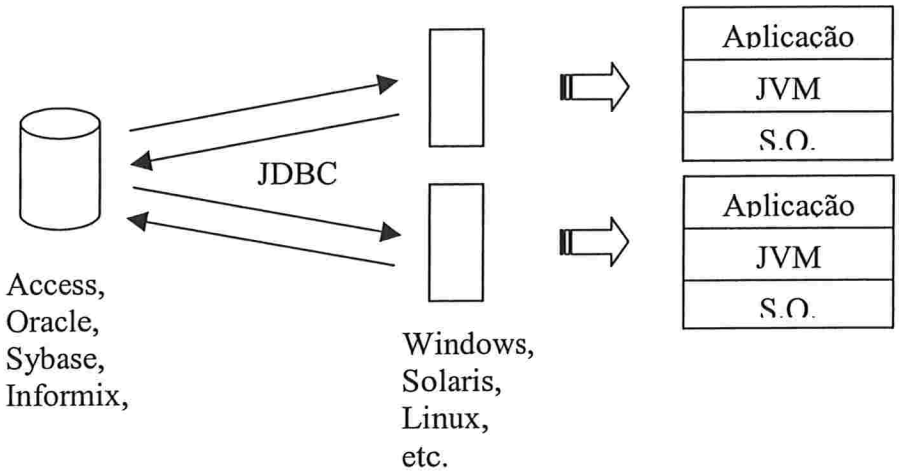
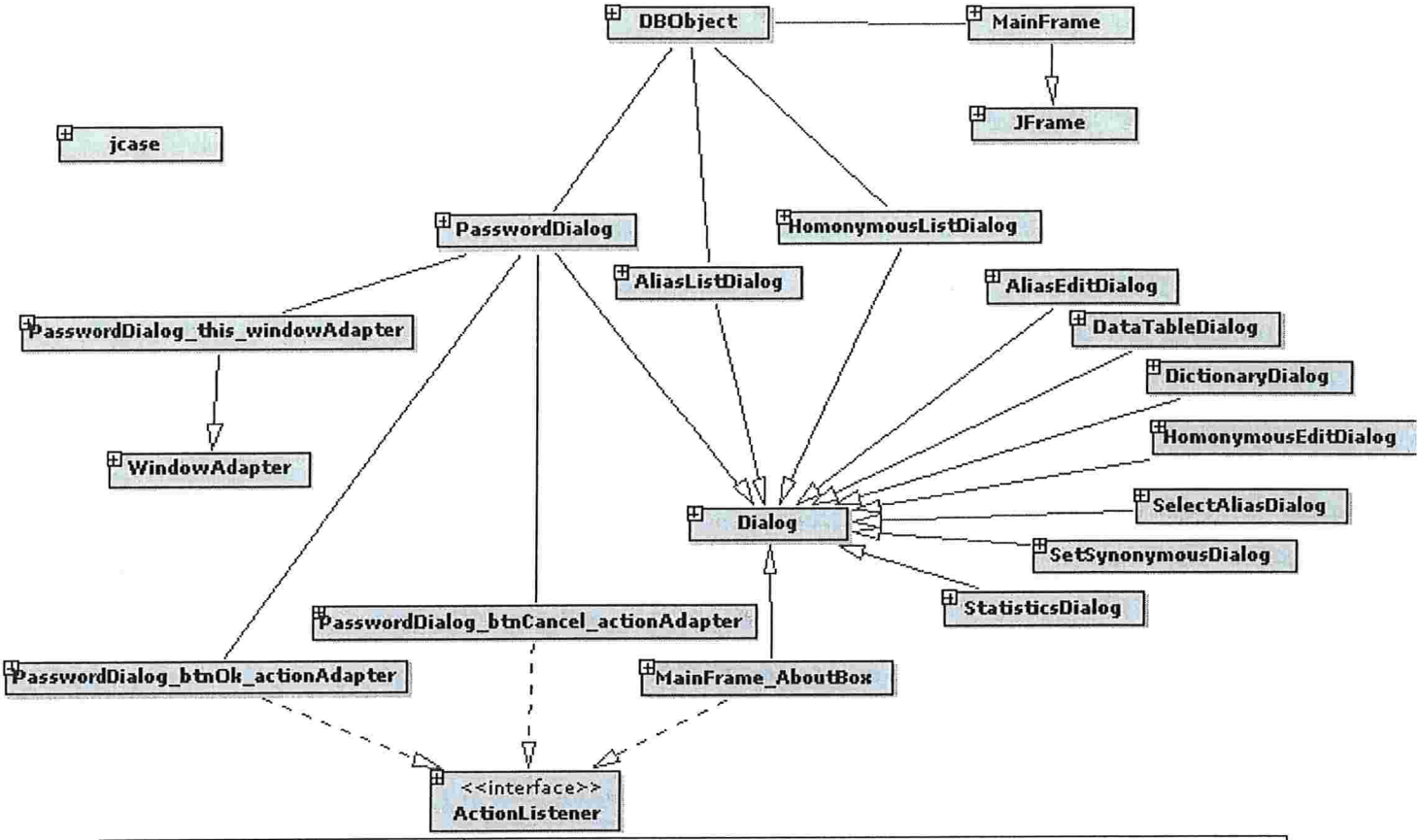


Fig. 7.4 – *Arquitetura adotada*

No caso, para facilitar o desenvolvimento e por ser uma ferramenta mono-usuário, optou-se por utilizar o banco de dados Access e o driver JDBC padrão da sun que utiliza o ODBC sun.jdbc.odbc.JdbcOdbcDriver. Embora este driver seja mais lento que um driver JDBC nativo para o banco de dados, permite acessar qualquer banco de dados que possua um driver ODBC.

7.6.2 Diagrama de Classes

A figura a seguir, representa o diagrama de classes da ferramenta implementada, para simplificar o diagrama e torná-lo compreensível, foram representadas apenas as classes implementadas e que possuem relevância para este estudo, não estando presentes as classes dos pacotes de componentes da sun utilizadas principalmente para tratamento de eventos e apresentação gráfica.



Classe	Funcionalidade
Jcase	Inicializa a aplicação
DBObject	Responsável por todo acesso a base de dados
MainFrame	Responsável pelo controle da aplicação
PasswordDialog	Tela de parametrização de conexão a base de dados
AliasListDialog	Tela que lista os apelidos
HomonymousListDialog	Tela que lista os homônimos
AliasEditDialog	Tela de edição de um apelido
DataTableDialog	Tela que exibe o resultado de uma consulta ao banco de dados
DictionaryDialog	Tela responsável pela atribuição de sinônimos/homônimos
HomonymousEditDialog	Tela de edição de um homônimo
SelectAliasDialog	Tela que lista ocorrências de um sinônimo
SetSynonymousDialog	Tela que atribui sinônimos
StatisticsDialog	Tela que exibe os valores estatísticos
MainFrame_AboutBox	Tela de informação da aplicação

Para a implementação desta ferramenta, adotou-se a abordagem *MVC - Model, View* and *Controll*, ou *Modelo*, *Apresentação* e *Controle*, que divide as classes implementadas em três tipos, de acordo com qual operação cada uma delas é responsável:

Classes de Apresentação – são as classes responsáveis por exibir e/ou obter dados de entrada para a aplicação. São classes de apresentação:

- `PasswordDialog`
- `AliasListDialog`
- `HomonymousListDialog`
- `AliasEditDialog`
- `DataTableDialog`
- `DictionaryDialog`
- `HomonymousEditDialog`
- `SelectAliasDialog`
- `SetSynonymousDialog`
- `StatisticsDialog`
- `MainFrame_AboutBox`

Classes de Modelo – são as classes responsáveis por obter e persistir dados do modelo e estruturas da aplicação. São classes de modelo:

- `DBObject`

Classes de Controle – são as classes responsáveis por controlar a execução da aplicação, recebendo e delegando requisições das classes de apresentação e das classes de modelo. São classes de controle:

- `Jcase`
- `MainFrame`

Com isto, é válido observar que toda a lógica da ferramenta está presente nas classes de controle, mais precisamente na classe `MainFrame`. No próximo item, será detalhada a lógica de obtenção dos valores utilizados no cálculo dos indicadores fornecidos pela ferramenta, que é controlada pela classe `MainFrame`.

7.6.3 Tabelas, Tabelas Temporárias e Consultas

Os dados se encontram armazenados em apenas uma tabela chamada `fields`, constituída pelos campos:

`Table_name` – nome da tabela que possui o campo
`Field_name` – nome do campo
`Datatype` – tipo de dado do campo
`Length` – tamanho do campo
`Precision` – precisão do campo
`Alias` – apelido do campo, segundo o significado

Para obter as demais informações, foram utilizadas consultas diretas sobre a tabela `fields` ou consultas sobre tabelas temporárias geradas a partir de consultas sobre a tabela `fields`. Caso o banco de dados suportasse, seria possível gerar view's da tabela `fields` que substituiriam as tabelas temporárias, o que tornaria a performance melhor, como o banco de dados escolhido não possui tal recurso, foram geradas as tabelas temporárias em tempo de execução.

O número de aliases foi obtido através das seguintes operações:

```
SELECT DISTINCT alias INTO tmp_aliases FROM fields
SELECT COUNT(*) FROM tmp_aliases
```

O número total de tabelas foi obtido através das operações:

```
SELECT DISTINCT table_name INTO tmp_tables FROM fields
SELECT COUNT(*) FROM tmp_tables
```

O número total de campos analisados:

```
SELECT COUNT(*) FROM fields
```

O número de campos distintos:

```
SELECT DISTINCT field_name INTO tmp_distinct_fields FROM fields
SELECT COUNT(*) FROM tmp_distinct_fields
```

O número total de campos com domínio divergente:

```
SELECT DISTINCT alias, datatype, length, precision
  INTO tmp_diff_domain FROM fields
SELECT alias, COUNT(*)
  INTO tmp_alias_diff_domain
  FROM tmp_diff_domain
  GROUP BY alias
  HAVING COUNT(*) > 1
SELECT COUNT(*) FROM tmp_alias_diff_domain
```

O número total de campos com divergência de tipo de dado:

```
SELECT DISTINCT alias, datatype
  INTO tmp_diff_domain FROM fields
SELECT alias, COUNT(*)
  INTO tmp_alias_diff_domain
  FROM tmp_diff_domain
  GROUP BY alias
  HAVING COUNT(*) > 1
SELECT COUNT(*) FROM tmp_alias_diff_domain
```

O número total de campos com divergência de tamanho de dado:

```
SELECT DISTINCT alias, length
  INTO tmp_diff_domain FROM fields
SELECT alias, COUNT(*)
  INTO tmp_alias_diff_domain
  FROM tmp_diff_domain
  GROUP BY alias
  HAVING COUNT(*) > 1
SELECT COUNT(*) FROM tmp_alias_diff_domain
```

O número total de campos com divergência de precisão de dado:

```
SELECT DISTINCT alias, precision
  INTO tmp_diff_domain FROM fields
SELECT alias, COUNT(*)
  INTO tmp_alias_diff_domain
  FROM tmp_diff_domain
  GROUP BY alias
  HAVING COUNT(*) > 1
SELECT COUNT(*) FROM tmp_alias_diff_domain
```

O número de campos homônimos:

```
SELECT DISTINCT alias, field_name
  INTO tmp_aliases FROM fields
SELECT field_name, COUNT(*)
  INTO tmp_homonymous
  FROM tmp_aliases
  GROUP BY field_name
  HAVING COUNT(*) > 1
SELECT COUNT(*) FROM tmp_homonymous
```

O número de campos sinônimos:

```
SELECT DISTINCT alias, field_name
  INTO tmp_aliases FROM fields
SELECT alias, COUNT(*)
  INTO tmp_synonymous
  FROM tmp_aliases
  GROUP BY alias
  HAVING COUNT(*) > 1
SELECT COUNT(*) FROM tmp_synonymous
```

Como é possível observar, tanto as consultas como os dados são simples e fáceis de serem obtidos. A dificuldade do processo encontra-se na análise dos campos e definição dos aliases, principalmente por ser um processo manual. Para facilitar este processo, é necessária uma série de funcionalidades, algumas delas serão discutidas no próximo item.

7.7 Características da Aplicação

A primeira necessidade observada é a de realizar a carga inicial dos dados, ou seja, carregar as informações sobre tabelas, campos e tipos de dados dos campos para a tabela `fields`. Para isto são necessárias as funcionalidades de importação e exportação de dados, detalhes desta funcionalidade são tratados nos próximos itens.

Para manter a consistência das informações e facilitar o trabalho, assume-se de início que campos com o mesmo nome possuem o mesmo significado, sendo portanto sinônimos. Ao ser realizada a carga inicial dos dados através da importação, o apelido de cada campo será inicializado com o próprio nome do campo.

Em seguida, são necessárias funções que permitam agregar ou desagregar apelidos de campos, ou seja, é necessário unificar apelidos que embora distintos possuam o mesmo significado e, por outro lado, desagregar em apelidos distintos, os campos que possuam significados distintos. É importante observar que estas operações manipulam apelidos e não campos individuais, portanto se dois apelidos, cada um com dezenas de campos são unificados, todos os campos dos dois apelidos serão associados a apenas um apelido. Esta operação poupa esforço de alterar individualmente os apelidos dos campos, embora a função de realizar a alteração individual dos apelidos dos campos também exista.

Outra operação implementada devido à sua grande funcionalidade é a sugestão de homônimos e sinônimos, através de algumas estratégias percebidas ao realizar as operações anteriores de forma manual, permitindo semi-automatizar os agrupamentos e desagrupamentos de apelidos. Neste caso, a ferramenta adota as seguintes estratégias para sugerir prováveis homônimos e sinônimos, que só serão efetivados após a confirmação do usuário:

- *Homônimos* – campos com mesmo nome, mas com tipos de dados distintos, são candidatos a campos homônimos. Por exemplo:

Tabela	Campo	Tipo de Dado	Tamanho	Precisão	Apelido
Tabela1	A	Texto	6		A
Tabela3	A	Número	6		A

Fig. 7.5 – *Exemplo de candidatos a homônimos*

No exemplo acima, os campos são candidatos a homônimos, pois possuem o mesmo nome embora os tipos de dados sejam diferentes, mas a decisão baseada na análise do significados de cada campo, é do usuário.

- *Sinônimos* – campos em que o nome de um, corresponde ao prefixo ou sufixo do nome de outro campo, são candidatos a sinônimos. Por exemplo:

Tabela	Campo	Tipo de Dado	Tamanho	Precisão	Apelido
Tabela1	B	Texto	6		B
Tabela2	B_1	Número	6		B_1
Tabela3	New_B	Texto	6		New_B

Fig. 7.6 – *Exemplo de candidatos a sinônimos*

No exemplo acima, os campos são candidatos a sinônimos, pois possuem parte de seus nomes em comum, esta característica assim como a anterior foi percebida após a análise de alguns casos. Da mesma forma, a sugestão de sinônimos pode ser aplicada utilizando o nome do apelido ao invés do nome do campo.

Uma funcionalidade não implementada, mas de grande utilidade seria a geração de uma base de equivalência de nomes ou palavras, onde seria possível cadastrar palavras ou códigos que possuem o mesmo significado, por exemplo a palavra CODIGO nos nomes dos campos das tabelas, pode ser substituída pela sigla CD. Portanto os campos X_CODIGO e X_CD são prováveis sinônimos, assim como AMOUT pode ser abreviado para AMT, etc.

7.7.1 Entrada de Dados

A entrada de dados é realizada a partir da importação de um arquivo texto respeitando a formatação, onde na primeira linha há a descrição dos nomes dos campos separados por tabulação: `table_name`, `field_name`, `datatype`, `length`, `precision` e `alias`; as linhas subseqüentes possuirão os dados referentes a cada coluna separados por tabulação.

Caso a coluna `alias` não seja informada, o sistema irá criar um alias idêntico ao nome do campo, desta forma, assume-se que cada ocorrência de um mesmo nome de campo indica que os mesmos são sinônimos. Por exemplo, caso o arquivo possua os seguintes dados:

Tabela	Campo	Tipo de Dado	Tamanho	Precisão
Tabela1	A	Texto	6	
Tabela1	B	Texto	2	
Tabela2	B	Texto	2	
Tabela2	C	Texto	12	
Tabela3	A	Número	6	
Tabela3	D	Número	12	2

Fig. 7.7 – *Modelo da tabela de entrada*

Serão geradas na base de dados do aplicativo as seguintes informações:

Tabela	Campo	Tipo de Dado	Tamanho	Precisão	Apelido
Tabela1	A	Texto	6		A
Tabela1	B	Texto	2		B
Tabela2	B	Texto	2		B
Tabela2	C	Texto	12		C
Tabela3	A	Número	6		A
Tabela3	D	Número	12	2	D

Fig. 7.8 – *Modelo da interpretação da tabela de entrada*

Através da análise das informações, do nome dos campos e do conhecimento sobre o modelo de dados, o usuário informará os sinônimos e homônimos contando com o auxílio da aplicação. Ë possível alterar diretamente o apelido de um campo, atribuir que todos os campos de um conjunto de apelidos assumam o valor de outro apelido, realizando assim um agrupamento de apelidos. O usuário pode ainda, analisar cada ocorrência de um apelido, listando todos os campos associados a ele, de modo a verificar se são realmente sinônimos ou homônimos.

7.7.2 Saída de Dados

Como resultado, o aplicativo irá gerar listagens de tabelas, campos, apelidos, sinônimos, homônimos, diferenças de domínio e ocorrências de apelidos, que podem ser utilizadas ainda na fase de agrupamento de aliases para auxiliar na identificação de sinônimos e homônimos.

O aplicativo gera também valores quantitativos que podem auxiliar a comparação e análise dos sistemas, fornecendo informações que permitam comparar tamanho e complexidade dos sistemas, como:

- *Número de Tabelas* – quantidade de tabelas analisadas;
- *Número de Campos* – apresenta a quantidade de nome de campos distintos e a quantidade absoluta de campos analisados;
- *Número de Apelidos* – quantidade de apelidos distintos atribuídos aos campos analisados;
- *Número de Sinônimos* - quantidade de sinônimos distintos, total de incidências e percentual de apelidos que são sinônimos;
- *Número de Homônimos* - quantidade de homônimos distintos, total de incidências e percentual de apelidos que são homônimos;
- *Número de Diferenças de Domínio* - quantidade de apelidos que possuem domínios diferentes e percentual de apelidos que possuem domínios distintos;
- *Número de Diferenças de Tipo de Dados* - quantidade de apelidos que possuem tipo de dados diferentes e percentual de apelidos que possuem esta diferença;

- *Número de Diferenças Tamanho de Dados* - quantidade de apelidos que possuem tamanho de dados diferentes e percentual de apelidos que possuem esta diferença;
- *Número de Diferenças Precisão de Dados* - quantidade de apelidos que possuem precisão de dados diferentes e percentual de apelidos que possuem esta diferença.

Estes valores serão utilizados no cálculo dos valores dos indicadores de qualidade propostos no capítulo 5.

7.8 Conclusão

Neste capítulo, foi exibida a lógica adotada para a construção da ferramenta de auxílio a obtenção dos valores dos indicadores definidos no capítulo 5. Neste sentido, foram definidos os conceitos de apelidos, de sinônimos e homônimos, de diferenças de domínio e de aspectos de segurança e rastreabilidade adotando a definição de apelido. Foi mostrada também, a arquitetura escolhida para a implementação da ferramenta, suas características e sua lógica implementada.

No próximo capítulo, serão mostrados os valores obtidos através do uso da ferramenta implementada para alguns casos de uso. Será também realizada uma análise dos valores dos indicadores obtidos.

Capítulo 8

Estudo de Casos

"Estudo de casos tratam de experiências reais na aplicação de sistemas onde são identificados problemas e oportunidades existentes" [Centro de Estudos em Logística].

Como parte deste estudo, foi realizada a análise de alguns sistemas através da ferramenta implementada, cujas características e conceitos foram definidos na capítulo anterior. Para isto foi gerado um arquivo texto, conforme a formatação descrita no capítulo anterior, e importado para o aplicativo de análise. Em seguida foi realizado um trabalho de análise das informações, agrupando-se os homônimos e sinônimos manualmente. Com isso, foi possível obter algumas informações como resultado que foram utilizadas no cálculo dos valores dos indicadores de qualidade definidos no capítulo 5.

A seguir serão expostos quatro estudos de casos de sistemas que utilizam bancos de dados relacionais:

- Sistema NorthWind – fornecido pela Microsoft junto com o banco de dados Access, como exemplo de implementação de banco de dados. Como será observados nos dados obtidos, se trata de um sistema de tamanho pequeno com poucas tabelas.
- Sistema de Pedido Eletrônico para Empresa de Alimentação – sistema desenvolvido para solicitação de pedido de benefício de alimentação de funcionários para empresa de alimentação. Como será observado, também se trata de um sistema pequeno, com poucas tabelas.
- Sistema de Telecomunicações I – sistema desenvolvido para gestão de uma empresa de telecomunicações. Entre os quatro estudos de casos, este sistema é o maior, apresentando uma quantidade de tabelas bem superior aos outros casos.
- Sistema de Telecomunicações II – sistema desenvolvido para gestão de clientes de outra empresa de telecomunicações, apresentando tamanho médio, com uma quantidade de tabelas maior que os sistemas NorthWind e de Pedido Eletrônico para Empresa de Alimentação, mas bem menor que o sistema de Telecomunicações I.

Nos casos dos sistemas de Pedido Eletrônico para Empresa de Alimentação, de Telecomunicações I e de Telecomunicações II, os dados utilizados são informações reais de sistemas utilizados comercialmente e fornecidos pelas empresas na condição de que fosse mantido o sigilo da fonte, ou seja da empresa, e que o conteúdo original não fosse divulgado, apenas os resultados.

8.1 Estudo de Caso – Sistema NorthWind

Foi analisada a implementação do banco de dados de exemplo do Access, o Northwind. Sendo este um sistema pequeno de poucas tabelas, os resultados obtidos foram:

Informações	Valores Distintos	Valores Totais	Percentual
Tabelas:	7	-	-
Campos:	51	73	-
Apelidos:	43	-	-
Homônimos:	0	0	0%
Sinônimos:	8	16	18,6%
Diferença de Domínio:	0	-	0%
Diferença de Tipo de Dado:	0	-	0%
Diferença de Tamanho de Dado:	0	-	0%
Diferença de Precisão:	0	-	0%

Fig. 8.1.1 – *Dados obtidos do sistema NorthWind*

Elemento	Fator	Medida
Taxa de Homônimos	0 / 43	0 %
Taxa de Sinônimos	8 / 43	18,6 %
Taxa de Divergência de Domínios	0 / 43	0 %
Taxa Média de Ausência de Informações de Segurança e Rastreabilidade	35 / 35	100%

Fig. 8.1.2 – *Resultados obtidos do sistema NorthWind*

Elemento	Fator	Medida
Taxa de Ausência de Identificação de Usuário Cadastrante	7 / 7	100%
Taxa de Ausência da Data de Cadastro	7 / 7	100%
Taxa de Ausência de Identificação de Usuário da Última Alteração	7 / 7	100%
Taxa de Ausência da Data da Última Alteração	7 / 7	100%
Taxa de Ausência de Identificação da Aplicação da Última Alteração	7 / 7	100%

Fig. 8.1.3 – *Resultados obtidos do sistema NorthWind*

A análise dos resultados revela que houve uma preocupação, no momento da implementação, em evitar a duplicação de significados para o mesmo nome de campo, ou seja, evitou-se a ocorrência de homônimos; também é possível concluir que os domínios de dados foram respeitados, pois não houve nenhuma ocorrência de divergência de tipos, tamanhos ou precisão de dados.

O único resultado que exigiu alguma análise mais detalhada foi a ocorrência de sinônimos. Embora a incidência tenha sido alta, 18,6% dos apelidos da base de dados representam sinônimos, 6 deles, ou 75% dos sinônimos, foram ocasionados por auto-relacionamentos ou tabelas de auto-relacionamento. Por exemplo, na tabela FUNCIONARIOS, o campo CÓDIGO_DO_FUNCIONÁRIO é o identificador único de cada funcionário, e cada funcionário está associado a seu respectivo supervisor através do campo SUPERVISOR, que contém o código do funcionário de seu supervisor. Desta forma, os campos SUPERVISOR e CÓDIGO_DO_FUNCIONÁRIO embora representem a mesma informação, ou seja, o código de identificação de um funcionário, possuem nome de campo diferentes, mas isto não representa um problema ou falta de atenção com a nomenclatura dos campos da tabela, pois os banco de dados relacionais não permitem dois campos com mesmo nome na mesma tabela. Os dois outros casos de sinônimos são realmente casos que representam campos que poderiam possuir o mesmo nome, mas possuem nomes distintos.

Este modelo de dados não possuía nenhum campo com informações de segurança e rastreabilidade, indicando um problema caso se deseje identificar o usuário ou aplicativo que realizou uma determinada operação no sistema.

8.2 Estudo de Caso – Sistema de Pedido Eletrônico para Empresa de Alimentação

Neste estudo de caso, foi analisado o modelo de dados de um sistema real de solicitação de pedidos de produtos de alimentação. Embora o sistema seja relativamente pequeno, os resultados obtidos serviram para alterar o modelo a fim de evitar problemas que poderiam ser ocasionados pelas divergências de tipos de dados. Os valores obtidos foram:

Informações	Valores Distintos	Valores Totais	Percentual
Tabelas:	27	-	-
Campos:	145	256	-
Apelidos:	122	-	-
Homônimos:	0	0	0%
Sinônimos:	15	38	12,3%
Diferença de Domínio:	3	-	2,46%
Diferença de Tipo de Dado:	1	-	0,82%
Diferença de Tamanho de Dado:	2	-	1,64%
Diferença de Precisão:	0	-	0%

Fig. 8.2.1 – *Dados obtidos do sistema de pedido eletrônico*

Elemento	Fator	Medida
Taxa de Homônimos	0 / 122	0 %
Taxa de Sinônimos	15 / 122	12,3 %
Taxa de Divergência de Domínios	3 / 122	2,46 %
Taxa Média de Ausência de Informações de Segurança e Rastreabilidade	116 / 135	85,93%

Fig. 8.2.2 – *Resultados obtidos do sistema de pedido eletrônico*

Elemento	Fator	Medida
Taxa de Ausência de Identificação de Usuário Cadastrante	18 / 27	66,67%
Taxa de Ausência da Data de Cadastro	27 / 27	100%
Taxa de Ausência de Identificação de Usuário da Última Alteração	27 / 27	100%
Taxa de Ausência da Data da Última Alteração	17 / 27	62,96%
Taxa de Ausência de Identificação da Aplicação da Última Alteração	27 / 27	100%

Fig. 8.2.3 – *Resultados obtidos do sistema de pedido eletrônico*

Assim como no caso anterior, não houve nenhuma ocorrência de homônimos, contudo houve a ocorrência de três casos de divergência de domínio. Um dos casos foi em relação ao tipo de dado, no caso o código de identificação do endereço, CD_END, que ora foi representado com o tipo Inteiro, ora com o tipo Inteiro Longo. Os outros dois casos foram ocasionados pela divergência no tamanho do dado representado: o campo nome da tabela NM_TAB que ora foi definido como Texto de tamanho 20, ora como Texto de tamanho 25 e o campo texto do complemento de endereço, TE_COMPL, que ora foi representado como Texto de 50 e ora como Texto de 60.

Este modelo de dados possui dois campos com informações de rastreabilidade e segurança: o código do usuário que realizou a última alteração, CD_USU, e a data da última atualização, DT_ULT_ALT. Enquanto o primeiro estava presente em 18 das 27 tabelas, o segundo estava presente em 17 tabelas, com isto foi possível determinar que, respectivamente 9 e 10 tabelas não possuíam estas informações quando deveriam.

8.3 Estudo de Caso – Sistema de Telecomunicações I

Neste estudo de caso, foi analisado o modelo de dados de um sistema real de uma empresa de Telecomunicações, sendo este o estudo que apresentou o maior volume de informações. Após a realização do processamento das informações, foram obtidos os seguintes valores:

Informações	Valores Distintos	Valores Totais	Percentual
Tabelas:	1151	-	-
Campos:	5355	24355	-
Apelidos:	4845	-	-
Homônimos:	9	31	0,19%
Sinônimos:	119	651	2,46%
Diferença de Domínio:	133	-	2,75%
Diferença de Tipo de Dado:	71	-	1,47%
Diferença de Tamanho de Dado:	88	-	1,82%
Diferença de Precisão:	41	-	0,85%

Fig. 8.3.1 – *Valores obtidos do sistema de Telecomunicações I*

Elemento	Fator	Medida
Taxa de Homônimos	9 / 4845	0,19 %
Taxa de Sinônimos	119 / 4845	2,46 %
Taxa de Divergência de Domínios	133 / 4845	2,75 %
Taxa Média de Ausência de Informações de Segurança e Rastreabilidade	608 / 5755	10,56 %

Fig. 8.3.2 – *Resultados obtidos do sistema de Telecomunicações I*

Elemento	Fator	Medida
Taxa de Ausência de Identificação de Usuário Cadastrante	130 / 1151	11,29%
Taxa de Ausência da Data de Cadastro	86 / 1151	7,47 %
Taxa de Ausência de Identificação de Usuário da Última Alteração	130 / 1151	11,29 %
Taxa de Ausência da Data da Última Alteração	130 / 1151	11,29 %
Taxa de Ausência de Identificação da Aplicação da Última Alteração	132 / 1151	11,47 %

Fig. 8.3.3 – *Resultados obtidos do sistema de Telecomunicações I*

Houve a ocorrência de 9 homônimos, representando 0,19% da quantidade de apelidos identificados, sendo este um valor muito baixo, comparado com o número de campos do modelo. O número de ocorrências de sinônimos e diferenças de domínio, embora bem maiores, 119 e 133 respectivamente, representam uma pequena proporção dos campos do modelo, 2,46% e 2,75%.

Este modelo apresenta vários campos contendo informações de segurança e rastreabilidade, sendo eles: SYS_CREATION_DATE, APPLICATION_ID, OPERATOR_ID, SYS_UPDATE_DATE, DL_SERVICE_CODE, DL_UPDATE_STAMP. O primeiro mostrou-se ausente em 86 das 1.151 tabelas e os demais em aproximadamente 130 tabelas.

8.4 Estudo de Caso – Sistema de Telecomunicações II

Foi analisado parte do modelo de dados de um sistema real de outra empresa de Telecomunicações, este modelo de dados refere-se apenas a alguns módulos do sistema de mesmo contexto. Este caso, embora não seja tão grande como o anterior, é o que mais se utilizaria dos resultados obtidos. Foram obtidos os seguintes resultados:

Informações	Valores Distintos	Valores Totais	Percentual
Tabelas:	73	-	-
Campos:	319	633	-
Apelidos:	303	-	-
Homônimos:	10	61	3,3%
Sinônimos:	31	98	10,23%
Diferença de Domínio:	23	-	7,59%
Diferença de Tipo de Dado:	10	-	3,3%
Diferença de Tamanho de Dado:	23	-	7,59%
Diferença de Precisão:	1	-	0,33%

Fig. 8.4.1 – *Resultados obtidos do sistema de Telecomunicações II*

Elemento	Fator	Medida
Taxa de Homônimos	10 / 303	3,3 %
Taxa de Sinônimos	31 / 303	10,23 %
Taxa de Divergência de Domínios	23 / 303	7,59 %
Taxa Média de Ausência de Informações de Segurança e Rastreabilidade	299/365	81,91 %

Fig. 8.4.2 – *Resultados obtidos do sistema de Telecomunicações II*

Elemento	Fator	Medida
Taxa de Ausência de Identificação de Usuário Cadastrante	53 / 73	72,60 %
Taxa de Ausência da Data de Cadastro	55 / 73	75,34 %
Taxa de Ausência de Identificação de Usuário da Última Alteração	59 / 73	80,82 %
Taxa de Ausência da Data da Última Alteração	59 / 73	80,82 %
Taxa de Ausência de Identificação da Aplicação da Última Alteração	73 / 73	100 %

Fig. 8.4.3 – *Resultados obtidos do sistema de Telecomunicações II*

Neste estudo de caso, ocorreu a maior incidência de homônimos, 10, representando 3,3% do total de apelidos atribuídos aos campos. O número de ocorrências de sinônimos e diferenças de domínio, também foram maiores do que os casos anteriores, com 10,23% e 7,59%.

Foram identificados os seguintes campos de informações sobre segurança e rastreabilidade: CUID, CDATE, CTIME, MUID, MDATE e MTIME, com respectivamente: 53, 55, 55, 59, 59 e 59 ausências em 73 tabelas do modelo. Os valores obtidos são considerados altos, caracterizando uma deficiência no rastreamento de usuários e aplicativos que manipularam as informações armazenadas.

8.5 Conclusão

Nos itens anteriores, foram exibidos os resultados obtidos através da utilização da ferramenta de auxílio à análise de qualidade, descrita no capítulo anterior, para cada caso de uso. Através dos valores obtidos, foram calculados os valores dos indicadores de qualidade estabelecidos no capítulo 5, e foi realizada também, uma análise individual da qualidade de implementação do banco de dados relacional utilizado em cada casos estudado. A seguir, no próximo capítulo, será realizada uma análise comparativa dos valores dos indicadores obtidos e serão apresentadas as conclusões obtidas com este estudo.

Capítulo 9

Conclusões

"Os problemas são oportunidades para se demonstrar o que se sabe" - Duke Ellington, músico de jazz americano, 1899-1974

Neste capítulo, serão expostas as conclusões obtidas tanto da implementação da ferramenta descrita no capítulo 7, quanto dos resultados obtidos da utilização desta ferramenta nos estudos de caso do capítulo 8.

9.1 Conceitos

Neste trabalho, buscou-se utilizar conceitos já existentes na literatura das diversas áreas, como é o caso dos conceitos de qualidade, de indicadores de qualidade, de engenharia de software, de desenvolvimento de software, de banco de dados relacionais, de modelo entidade-relacionamento, etc.

O único conceito mais específico definido neste trabalho é o conceito de apelidos, mesmo assim, sua definição é bem simples. Apelido corresponde a um código que identifica univocamente o significado semântico de cada informação envolvida em um sistema. É utilizado para agrupar campos segundo o significado da informação representada. Da definição de apelidos, derivam as definições de sinônimos e homônimos, e é sobre os apelidos que se realiza a obtenção dos valores dos indicadores. Por exemplo, as divergências de domínio e meta-informações de segurança e rastreabilidade são sempre obtidas avaliando-se os apelidos dos campos e não seus nomes.

Na verdade, a idéia é que cada campo do sistema analisado possui um relacionamento de cardinalidade 1:1 como seu significado, portanto, cada campo está associado a apenas um significado semântico, por isso a escolha por adicionar a coluna `alias` na tabela `field`, para armazenar a informação do significado semântico de cada campo na ferramenta de auxílio implementada.

Conforme o esperado, os campos que contém meta-informações de segurança e rastreabilidade, foram os campos de maior incidência nas tabelas dos sistemas.

Como foi dito anteriormente, o modelo de dados necessário é bem simples, composto basicamente por uma tabela, chamada `fields`, de onde todos os resultados são obtidos através de consultas. Muitas das consultas adotaram tabelas temporárias, que foram utilizadas como um artifício para que a consolidação de valores fosse realizada pelo banco de dados. Deste modo, foi possível simplificar a lógica da ferramenta, que apenas realiza as consultas e exibe os resultados, sem nenhum tratamento dos dados obtidos. Como conseqüência direta, o diagrama de classes da aplicação é bem simples e limpo.

9.2 Processo

A geração de um arquivo de importação para a realização da carga inicial das informações na aplicação, conforme descrito no item 7.7.1, para os casos 8.3 e 8.4, baseados respectivamente em Oracle e em um Banco de Dados proprietário, foi simples e fácil, sendo gerado automaticamente; já nos casos 8.1 e 8.2, que correspondem a sistemas baseados em Access, não há o recurso de se gerar este arquivo automaticamente, sendo necessário gerá-lo manualmente. Como em ambos os casos os sistemas não eram tão grandes, possuíam algumas dezenas de tabelas, foi possível realizar a tarefa de geração manual do arquivo, sendo contudo necessária muita atenção para traduzir o modelo do sistema para um arquivo, de modo a não gerar interferências na análise dos resultados ocasionadas por erros de mapeamento das informações.

As informações necessárias para a geração do arquivo, poderiam ser obtidas diretamente pela ferramenta implementada, tendo em vista que a linguagem java possui uma série de recursos que permitem obter informações do meta-modelo de um banco de dados, como é o caso da interface ResultSetMetaData implementada por objetos do pacote java.sql. Esta nova funcionalidade poderá ser implementada em uma nova versão da ferramenta.

Na seqüência do processo, com as informações na ferramenta implementada, foi necessário realizar as agregações e desagregações dos campos segundo seu significado semântico através da atribuição dos apelidos. Para realizar esta tarefa foi necessário um conhecimento do modelo do banco de dados maior do que o esperado, principalmente nos estudos de caso em que o volume de tabelas era grande tornou-se difícil a análise dos campos. Devido a esta dificuldade, surgiu a necessidade de mecanismos de auxílio à análise, com base em casos tratados manualmente, foi observada a possibilidade da implementação de algumas técnicas para a ferramenta sugerir prováveis homônimos e sinônimos, que foram realizadas e descritas no capítulo 7. Basicamente a estratégia para sugerir prováveis homônimos foi de listar campos que apesar de possuírem o mesmo nome, apresentam domínios diferentes; e para os sinônimos, campos cujo nome corresponda ao prefixo ou sufixo de outro campo.

Novamente, seria muito útil a existência de um banco de dados de equivalência de palavras conforme descrito em 7.7, que serviria de base para a ferramenta sugerir prováveis sinônimos, utilizando as estratégias descritas acima. Esta nova funcionalidade poderá ser implementada em uma nova versão da ferramenta.

Quanto à análise dos resultados obtidos, é possível tanto uma avaliação da qualidade individual de um sistema, analisando-se os valores obtidos mediante as metas esperadas, como uma avaliação comparativa entre os diversos sistemas, muito útil, por exemplo, na decisão de um fornecedor de serviços. Neste caso, o objetivo que era a avaliação da implementação do produto do processo de desenvolvimento do sistema de banco de dados relacional, e não do processo em si, foi alcançado.

9.3 Implementação da Ferramenta

Como era esperado, a utilização de uma linguagem orientada a objetos permitiu uma reutilização de código, reutilização e modularização dos componentes implementados. Por exemplo a classe DataTableDialog que exibe o resultado de uma consulta ao Banco de Dados, é utilizada por uma série de outros objetos da aplicação; já através da classe DBObject, foi possível isolar em um único objeto todo o acesso à base de dados. Quanto à reutilização de código, todas as classes implementadas, ou utilizam ou são derivadas de classes bases da biblioteca de classes de swing da sun, principalmente as classes gráficas, poupando assim esforço de implementação de objetos que não fazem parte do enfoque principal da aplicação.

Em termos de funcionalidade, a ferramenta conseguiu executar o seu papel, que é o de facilitar a manipulação dos dados e permitir a extração de informações necessárias para a análise dos indicadores de qualidade definidos e, consecutivamente, uma análise comparativa da qualidade dos diversos estudos de casos realizados. Alguns pontos de melhoria conforme do item anterior foram destacados, e poderão estar presentes em uma nova versão da ferramenta.

A portabilidade da ferramenta, como era esperado, é grande, podendo ser utilizada em diversos sistemas e hardwares, acessando vários tipos de banco de dados, principalmente porque o número de sistemas e banco de dados que dão suporte a tecnologia java tem aumentado continuamente, como é o caso de produtos como: Oracla, Sybase, Informix, DB2, etc. Quanto à performance, mesmo no caso de estudo de maior número de dados, que contava com dezenas de milhares de campos, e utilizando uma instância local do banco de dados Access, o tempo de resposta foi muito bom, obtendo os resultados de modo quase imediato, levando alguns segundos. Entretanto, por esta ferramenta ter uma função de análise, performance não é um aspecto crítico.

Conforme dito no item 9.1, o modelo de dados é simples, composto apenas por uma tabela e todos os resultados são obtidos através de consultas a esta tabela. A utilização de tabelas temporárias para a consolidação de valores, apesar de onerar o trabalho do banco de dados não prejudicou em muito a performance. A decisão por adotar este mecanismo foi com o objetivo de tornar a ferramenta bem simples e clara, como é possível observar no diagrama de classes.

9.4 Análise de Resultados

Em uma análise geral de todos os casos estudados, os resultados revelam que a ocorrência de sinônimos sempre foi mais alta que a de homônimos, em parte, a maioria das incidências de sinônimos foram ocasionadas por auto-relacionamentos, o que é justificável. Contudo, a presença de sinônimos apresentou-se como o indicador de freqüência mais alta tanto em sistemas menores como maiores, mostrando uma falta de preocupação por parte dos desenvolvedores em manter um dicionário de dados e tornando o entendimento do modelo mais complexo, acarretando os problemas e desvantagens já discutidos anteriormente no

capítulo 5. Os valores para a taxa de sinônimos foi de 18,6% para o caso 8.1, 12,3% para 8.2, 2,46% para 8.3 e 10,23% para 8.4.

Outro aspecto verificado foi com relação aos campos de meta-informação. Este tipo de informação, muitas vezes desprezado ou desnecessário em sistemas menores, pode ser vital em sistemas maiores, onde a administração de sua presença nas tabelas dos sistemas se torna mais difícil, exatamente pelo número de tabelas envolvidas. Aqui também este indicador apresentou valores significativos: a taxa média de ausência de informações de segurança e rastreabilidade foi de 85,93% das tabelas no caso 8.2, mais de 10% no caso 8.3 e 81,92% no caso 8.4. Justamente o sistema de maior porte, com o número de tabelas na ordem de milhares, foi o que apresentou maior preocupação e controle com este tipo de informação. Ao passo que o caso 8.1, justamente o menor, com um número tabelas na ordem de dezenas, apresentou ausência total deste tipo de informação, justificável apenas se o sistema for mono-usuário e não integrado a outros sistemas.

A informação obtida para quantificar este último indicador, permite até identificar quais tabelas não possuem esta informação, permitindo que alguma ação possa ser trabalhada no sentido de incluir estas informações, de modo a melhorar os valores para este indicador.

Em contrapartida, analisando-se os homônimos, evitar a sua ocorrência é mais facilmente controlada em sistemas menores. Em sistemas maiores, caso não haja uma política muito rígida em se estabelecer um dicionário de dados, e mesmo assim, dependendo do tamanho do sistema, é possível que homônimos ocorram. A taxa de homônimos obtida para cada um dos casos foi de: 0% para os casos 8.1 e 8.2; 0,19% para 8.3 e 3,3% para 8.4. Analisando estes valores, fica bem claro que em sistemas menores como 8.1 e 8.2, ambos com o número de tabelas na ordem de poucas dezenas, a ocorrência de homônimos é inexistente. Já no caso 8.4, que possui o número de tabelas na ordem de dezenas mas bem maior, o controle sobre a ocorrência de homônimos torna-se mais difícil, exigindo uma política mais rígida como é o caso de estabelecer um dicionário de dados. Em contrapartida, no caso 8.3, por ser bem maior mas apresentar apenas 0,19% de homônimos, fica clara a preocupação com o controle e utilização de um dicionário de dados.

Quanto ao indicador de diferença de domínio, este é o indicador que melhor apresenta falhas de implementação, devido a suas conseqüências como foi dito anteriormente no capítulo 5. A taxa de divergência de domínios alcançou os valores de 0% para o caso 8.1; 2,46% para o caso 8.2; 2,75% para 8.3 e 7,59% para 8.4. Observa-se aqui que uma incidência alta na maiorias dos casos, apenas em 8.1 não houve ocorrências, justamente por ser um sistema pequeno. Em 8.4, o número de ocorrências é muito alto comparado com 8.2 e 8.3, ficando claro a falta de qualidade nesta implementação.

Pode-se observar, através da análise dos indicadores, que o tamanho do sistema possui uma influência representativa nos valores obtidos e que, não necessariamente, um sistema muito grande tem uma qualidade inferior a outros bem menores. Porém é possível, e até mais fácil, gerar sistemas menores de pior qualidade, provavelmente pela ausência de políticas e preocupações com o desenvolvimento, que são vitais para sistemas de maior porte.

9.5 Aplicabilidade

A aplicabilidade da técnica descrita neste trabalho, bem como da ferramenta implementada, mostrou-se maior do que apenas para mensurar o grau de qualidade da implementação de uma solução adotando banco de dados relacionais. Pode ser aplicada também no auxílio ao processo de integração de bancos de dados heterogêneos, na migração de dados entre sistemas, data mining, em projetos de data warehouse e data mart, auxiliando em cada um deles, no entendimento e conhecimento de um banco de dados relacional já existente.

Este processo de obter um entendimento de um banco de dados já existente, a partir de seus dados, de suas dependências e implementação lógica, foi chamado de "*Data Profiling*" [DAMA 98]. O Data Profiling é composto por quatro elementos: Perfil de Domínio (Domain Profiling), Análise de Sinônimos/Homônimos, Análise de Dependências e Avaliação de Qualidade. Pode-se observar que este trabalho contempla três dos elementos deste processo: o Perfil de Domínio, que corresponde a extração das meta-informações do modelo dos sistemas e a definição de seus domínios; a Análise de Sinônimos/Homônimos, que corresponde a utilização do conceito de apelidos; e Avaliação de Qualidade, contida na análise dos resultados obtidos pela ferramenta implementada. A única abordagem não contemplada é a de Análise de Dependências que trata das dependências de informações, mas por ser necessário um conhecimento ainda maior do modelo conceitual e entendimento dos modelos, não foi utilizada neste trabalho.

Vale ressaltar a utilização prática da ferramenta implementada na identificação de falhas da implementação em um dos casos de uso, onde através da funcionalidade da ferramenta de listas as ocorrências, foi possível identificar e corrigir a divergência no domínio de alguns campos e a ausência de campos de meta-informação de segurança e rastreabilidade em algumas tabelas. É possível utilizar a ferramenta e a técnica descrita neste trabalho no acompanhamento do ciclo de vida de um mesmo sistema, permitindo analisar o impacto na qualidade do sistema a medida em que o sistema sofre alterações e é ampliado.

9.6 Trabalhos Futuros

Conforme o capítulo 5, foram definidos outros indicadores de qualidade pertencentes ao nível conceitual e do mapeamento de um relacionamento binário, que não foram tratados tanto pela ferramenta de auxílio a análise de qualidade, proposta no capítulo 7, como também não foram aplicados nos casos de uso, apresentados no capítulo 8. A ferramenta de auxílio à análise de qualidade pode evoluir para tratar estes indicadores, pode também, passar a realizar o processo de coleta dos dados necessários para a ferramenta diretamente dos sistemas analisados, sem a necessidade de geração de arquivos de entrada, conforme descrito em 9.2. Seria muito útil também, a existência de um banco de dados de equivalência de palavras, conforme descrito em 7.7, que facilitaria o processo de agrupamento de campos, sugerindo prováveis homônimos e sinônimos.

Eventualmente, outros indicadores de qualidade podem ser definidos que avaliem, por exemplo, banco de dados orientados a objetos, aspectos do nível físico, conceitual, de

mapeamento do nível conceitual para o lógico, ou do lógico para o físico. Com isso, outros conceitos e estratégias podem ser definidos.

A análise dos nomes dos objetos presentes no banco de dados pode ser ampliada e detalhada, levando-se em consideração os conceitos de:

- *Campo Semântico* – corresponde a área coberta, no domínio do significado, por uma palavra ou por um grupo de palavras da língua. Por exemplo, mesa: mesa de trabalho, mesa de eleição, mesa da assembléia, mesa de operação, pôr as cartas na mesa, etc. Pode-se estudar o campo semântico de um grupo de palavras como: dar, contar, desenhar, ordenar, etc., por serem verbos ligados por um elemento semântico comum;
- *Homógrafos* – palavras que apresentam a mesma escrita mas com significados distintos. Por exemplo: a palavra manga, que pode representar tanto a fruta como a parte do vestuário; jogo, substantivo, e jogo, verbo;
- *Polissemia* – corresponde a propriedade de uma palavra apresentar muitos significados. Por exemplo, a palavra banco pode possuir o sentido de assento, camada de pedra, elevação do fundo do mar, instituição financeira, etc.

Os nomes podem ser ainda analisados segundo seu grau de especialização/generalização, quando apresentam uma estreita relação entre eles:

- *Hipônimo* – é um vocábulo mais específico que outro. Por exemplo: rosa, dália, violeta, etc. são hipônimos. O hipônimo é sempre uma subcategoria de uma classe mais ampla: cachorro é um hipônimo de animal; vira-lata é um hipônimo de cachorro;
- *Hiperônimo* – é um vocábulo mais genérico que outro. Por exemplo: flor é um hiperônimo. Ao contrário do hipônimo, o hiperônimo representa a classe mais ampla e abrangente: animal é um hiperônimo de cachorro, de boi, de cavalo etc.

Como é possível observar um hiperônimo abrange seus hipônimos: flor que é um hiperônimo, e rosa, violeta, dália, etc., que são hipônimos. Estes termos são muito utilizados para definir a relação entre sinônimos, pois ao analisar os termos que se encontram em relação de sinonímia, verifica-se que, quase sempre, um deles, o hiperônimo, é mais abrangente que os demais, os hipônimos.

É possível então definir uma relação de hiponímia entre dois termos, ou hiperonímia pois depende apenas da forma como se analisa. Por exemplo, entre remoto e longínquo, longínquo se aplica a tudo o que está longe, enquanto que remoto, mais restrito, designa tudo que está longe e é de difícil acesso. Tudo o que é remoto é longínquo, mas nem tudo o que é longínquo é remoto, assim como todo cachorro é um animal, mas nem todo animal é um cachorro.

Com base neste conceitos é possível especificar ainda mais as relações entre os nomes dos objetos presentes num sistema de banco de dados, permitindo um análise ainda mais

detalhada dos homônimos e sinônimos, além de estabelecer uma hierarquia entre os significados semânticos dos nomes segundo seu grau de especialização/generalização.

Novamente, vale ressaltar, que a aplicabilidade destas análise é útil tanto na definição de fornecedores, como no acompanhamento dos índices de qualidade de um sistema ao longo de seu ciclo de vida.

Bibliografia

[Andersson 94] Martin Andersson, *Extrating an Entity Relationship Schema from a Relational Database trought Reverse Engineering*, Swiss Federal Institute of Technology, Department of Computer Science, In Proc. of the 13th Int. Conference of the Entity Relationship Approach, Manchester, Springer, 1994, pages 403-419.

[Batini, Ceri e Navathe 92] C. Batini, S. Ceri e S. Navathe, *Conceptual Database Design – An Entity-Relationship Approach*, The Benjamin/Cummings Publishing Companhy, Inc, 1992.

[Boehm 88] Barry W. Boehm, *A Spiral Model of Software Development and Enhancement*, IEEE Computer, Volume 21, Número 5, Maio de 1988, págs. 61-72.

[Chen 76] P. P. Chen, *The Entity-Relationship Model: Toward a Unified View of Data*, ACM Transactions on Data Base Systems, Volume 1, Número 1, Janeiro de 1976, págs. 9-36.

[Codd 70] E. F. Codd, *A Relational Model for Large Shared Data Banks*, Comunications of the ACM, Volume 13, Número 6, Junho de 1970, págs. 377-387.

[Crosby 94] Philip B. Crosby, *Qualidade é Investimento*, Rio de Janeiro, Ed. José Olympio, 1994.

[DAMA 98] DAMA – Chicago, February, 1998 – Newsletter, Jack Olson – VP. Engineering & Technology, DBStar, Inc. Online Edition: http://www.damachicago.org/feb98.htm

[Date 86] C. J. Date, *Introdução a Sistemas de Banco de Dados*, Ed. Campos, 4ª edição, Rio de Janeiro, 1986.

[Deming 90] W. Edward Deming, *Qualidade: A Revolução da Administração*, Rio de Janeiro, Ed. Marques-Saraiva, 1990.

[Elmasri e Navathe 89] R. Elmasri e S. Navathe, *Fundamentals of Database Systems*, The Benjamin/Cummings Publishing Company, 1989.

[Feingenbaum 86] A. V. Feingenbaum, *Total Quality Control, Engineering and Management*, New York, Ed. MC Graw-Hill, 1986.

[Freitas e Leite 97] Felipe Gouveia Freitas e Júlio César Sampaio do Prado Leite, *Aplicando Reuso de Software na Construção de Ferramentas de Engenharia Reversa*, Simpósio Brasileiro de Engenharia de Software (SBES), Pontifícia Universidade Católica do Rio de Janeiro, Departamento de Informática, 1997.

[Garvin 92] David A. Garvin, *Gerenciando a Qualidade: Visão estratégica e Competitiva*, Rio de Janeiro, Ed. Qualitymark, 1992.

[Gilb 93] Tom Gilb, *Principles of Software Engineering Management*, Addison Wesley, reimpressão, 1993.

[Gouzee, Mazijn e Billharz 95] N. Gouzee, Mazijn, B. & Billharz, *Indicators of Sustainable Development for Decision-Making*. Report of the Workshop of Ghent, Submitted to UN Comission on Sustainable Development. Published by the Federal Planning Office of Belgium, Belgium, págs. 9-11, January 1995.

[Heuser 99] Carlos A. Heuser, *Projeto de Banco de Dados*, Ed. Sagra Luzzatto, 2ª edição, Porto Alegre, 1999.

[Ishikawa 86] Kaoru Ishikawa, *TQC, Total Quality Control: Estratégia e Administração da Qualidade*, São Paulo, IMC Internacional Sistemas Educativos, 1986.

[Juran 92] J. M. Juran, *A Qualidade desde o Projeto: Novos Passos para o Planejamento de Qualidade em Produtos e Serviços*, São Paulo, Ed. Pioneira, 1992.

[Korth e Silberschatz 95] Henry F. Korth e Abraham Silberschatz, *Sistemas de Banco de Dados*, Ed. Makron Books , 2ª edição Revisada, 1995.

[McCracken e Jacson 82] D. D. McCracken and M. A. Jackson, *Life-Cycle Concept Considered Harmfull*, ACM Software Engineering Notes, págs. 29-32, April 1982.

[Mesquita 98] Eduardo José Soler Mesquita, *Projeto de Banco de Dados Distribuídos*, Tese de Mestrado, Instituto de Matemática e Estatística da USP, São Paulo, maio de 1998.

[Neild e Henschen 97] Tania Neild and Larry Henschen, *Managing Subjectivity in Data Integration*, Northwestern University, Department of ECE, Association for Information Systems, America Conference, Indianopolis, Indicane, August, 1997.

[Oliveira, Latelme e Formoso 95] Mírian Oliviera, Elvira Lantelme e Carlos Torres Formoso, *Sistema de Indicadores de Qualidade e Produtividade da Construção Civil - Manual de Utilização*, 2a. edição. Porto Alegre, junho 1995.

[Premerlani e Blaha 94] William J. Premerlani and Michael R. Blaha, *An Approach for Reverse Engineering of Relational Database*, Comunications of the ACM, Volume 37, Número 5, Maio de 1994, págs. 42-49.

[Pressman 95] Roger S. Pressman, *Engenharia de Software*, Ed. Makron Books, São Paulo, 1995.

[Rolt 98] Mírian Inês Pauli de Rolt, *O Uso de Indicadores para a Melhoria da Qualidade em Pequenas Empresas*, Tese de Mestrado, Universidade Federal de Santa Catarina, Florianópolis, junho de 1998.

[Royce 70] W. W. Royce, *Managing the Development of Large Software Systems: Concepts and Techniques*, in Proceedings WESCON, August, 1970.

[Setzer 89] Waldemar Setzer, *Banco de Dados: Conceitos, Modelos, Gerenciadores, Projeto Lógico, Projeto Físico*, Ed. Edgard Blucher, 3ª edição, São Paulo, 1989.

[Silva 99] Alécia Paula Mol e Silva, *Migrando Banco de Dados Relacionais para Tecnologia Objeto-Relacional*, SPG' 99, III Semana de Pós Graduação, universidade Federal de minas Gerais, Departamento de Ciência da Computação, junho de 1999.

[SMA 94] Secretariado para Modernização Administrativa, *Indicadores e Padrões de Qualidade*, Ministério da Reforma do Estado e da Administração Pública, Portugal, 1994.

[Ullman 88] Jeffrey D. Ullman, *Principles of Database and Knowledge-Base Systems*, Volume I, Ed. Computer Science Press, Rockville, 1988.