UNIVERSIDADE METODISTA DE PIRACICABA

FACULDADE DE CIÊNCIAS EXATAS E DA NATUREZA

MESTRADO EM CIÊNCIA DA COMPUTAÇÃO

ESTENDENDO UML PROFILE DE BANCO DE DADOS PARA PROJETO FÍSICO

EDSON LUIZ AVANZI

ORIENTADOR: PROF. DR. LUIZ CAMOLESI JR.

PIRACICABA, SP
2006

UNIVERSIDADE METODISTA DE PIRACICABA

FACULDADE DE CIÊNCIAS EXATAS E DA NATUREZA

MESTRADO EM CIÊNCIA DA COMPUTAÇÃO

ESTENDENDO UML PROFILE DE BANCO DE DADOS PARA PROJETO FÍSICO

EDSON LUIZ AVANZI

ORIENTADOR: PROF. DR. LUIZ CAMOLESI JR.

Dissertação apresentada ao Mestrado em Ciência da Computação, da Faculdade de Ciências Exatas e da Natureza, da Universidade Metodista de Piracicaba - UNIMEP, como requisito para obtenção do Título de Mestre em Ciência da Computação.

PIRACICABA, SP
2006

ESTENDENDO UML PROFILE DE BANCO DE DADOS PARA PROJETO FÍSICO

AUTOR: EDSON LUIZ AVANZI

ORIENTADOR: LUIZ CAMOLESI JR.

Dissertação de Mestrado defendida e aprovada em 23 de fevereiro de 2006, pela Banca Examinadora constituída pelos Professores:

Prof. Dr. Luiz Camolesi Jr, Presidente

UNIMEP

Prof. Dr. Luiz Eduardo Galvão Martins

UNIMEP

Prof. Dr. Carlos Roberto Valêncio

UNESP

DEDICATÓRIA

A Deus Criador de toda a humanidade que me tem presenteado com a melhor seleção de sua criação, meus pais, avós, irmãos, parentes, amigos e conhecidos.

Dedico a todos que, direta ou indiretamente, ajudaram-me para que esta etapa de minha vida pudesse ser vencida.

AGRADECIMENTOS

Nos caminhos mais difíceis que passei e nos propósitos mais firmes e enfáticos que empenhei é meu desejo agradecer a todos, contudo, sei que, injustificavelmente, esqueceria pessoas importantes. Por esta razão agradeço a Deus pelas pessoas maravilhosas que colocou em meu caminho.

Ao meu orientador e amigo Prof. Dr. Luiz Camolesi Jr. a quem muito admiro, pela sua compreensão, sabedoria e respeito às minhas idéias na orientação desta dissertação.

A UNIMEP e a todos os professores do PPGCC, pela competência, pelo nível de qualidade do curso e por todo conhecimento propiciado.

Aos meus pais, Antenor e Yolanda, pelos conselhos e incentivos, pilares para a minha formação.

A minha esposa, Sandra, pela sua paciência e entendimento da importância deste longo projeto.

Ao meu filho, Daniel, que chegou durante o período em que essa dissertação estava sendo redigida, renovando-me os ânimos para que pudesse ser concluída com sucesso.

“O que move os homens geniais não são as

novas idéias, mas a sua obsessão pela idéia de que

o que já foi dito ainda não é o suficiente”.

Eugène Delacroix

Pintor Francês

(1799-1863)

ESTENDENDO UML PROFILE DE BANCO DE DADOS PARA PROJETO FÍSICO

RESUMO

A engenharia de banco de dados é feita através de alguns contextos, desde o nível conceitual até o nível físico. Atualmente, muitas ferramentas estão disponíveis para auxiliar os projetistas, mas a UML (Linguagem de Modelagem Unificada) tem se tornado um padrão para descrever o desenvolvimento completo de bancos de dados relacional e relacional-objeto. Entretanto, alguns aspectos importantes sobre desempenho, no nível físico, não são suficientemente expressos na UML. Este trabalho tem como objetivo indicar aspectos como indexação, organização de arquivo e a configuração do servidor de banco de dados ao propor uma extensão do perfil UML de banco de dados para o projeto físico.

PALAVRAS-CHAVE: Linguagem de Modelagem Unificada, Projeto Físico de Banco de Dados

EXTENDING UML PROFILE OF DATABASE FOR PHYSICAL DESIGN

ABSTRACT

The database engineering is made through some contexts from the conceptual level to the physical level. Nowadays many tools are available to assist the designers with the projects but UML (Unified Modeling Language) has become a standard language to describe the complete development of relational and object relational databases. However, some important aspects about the performance of database systems, on the physical level, are not enough expressed in UML. This project has the objective to approach aspects such as indexing, file organization and the database server-computer configuration to propose an extension of the UML profile of database for physical design.

KEYWORDS: Unified Modeling Language, Database Physical Design

SUMÁRIO

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

AUSC	Análise no Uso dos Servidores Corporativos
BUC	Boletim de Utilização do Computador
CASE	Computer-Aided Software Engineering
CCM	Corba Component Model
CONDOR	Controle das Operações Rotinizadas
COSC	Controle das Operações nos Servidores Corporativos
DBA	Database Administrator
DCL	Data Control Language
DDL	Data Definition Language
DFP	Descrição do Fluxo de Processo
DML	Data Manipulation Language
DQL	Data Query Language
EAI	Enterprise Application Integration
EDOC	Enterprise Distributed Object Computing
ELA	Entity Link Attribute
HUTN	Human-Usable Textual Notation
ICAM	Integrated Computer-Aided Manufacturing
IDEF	ICAM Definition Language
JDBC	Java Database Connectivity
OMG	Object Management Group
OMT	Object Modeling Technique
OOSE	Object Oriented Software Engineering
OSTN	Object State Transition Network
PFD	Process Flow Description
QoS	Quality of Service
ROWID	Row Identificator
RTEO	Rede de Transição Estado-Objeto

SADT	Structured Analysis and Design Techniques
SGBD	Sistema Gerenciador de Banco de Dados
SQL	Structured Query Language
SQLJ	SQL-Java
UML	Unified Modeling Language
UOB	Unit Of Behavior

Capítulo 1. INTRODUÇÃO

1.1 CONSIDERAÇÕES INICIAIS

Os bancos de dados estão maiores e suportando, cada vez mais, tipos diferentes de dados. Isso tem ocasionado esquemas complexos que não satisfazem os requisitos de desempenho sem a degradação de acesso (DAVENPORT, 1998) e essa realidade vem sendo suprida por estratégias intuitivas e procedimentos empíricos de projeto físico de banco de dados pela engenharia de banco de dados (MCFADDEN, HOFFER & PRESCOTT, 1999).

A engenharia de banco de dados deve ser realizada considerando contextos que abordam desde o nível mais alto - representando a visão dos usuários sobre os dados, até o nível mais baixo - considerando o modo como os dados são armazenados e recuperados no banco de dados. Quanto maior o nível de abstração, maior a quantidade de detalhes do projeto, sendo que um nível mais baixo é produto de um nível mais alto (MULLER, 1999). Para a representação destas especificações pode ser utilizada a descrição visual, por meio do emprego de metáforas (MARTIN, 2002), especificando o complexo esquema do banco de dados e transformando a informação abstrata em expressiva e fácil informação visual (CATARCI, COSTABILE & MATERA, 1995).

A UML, Linguagem de Modelagem Unificada, é uma dessas linguagens, sendo utilizada para a especificação, visualização, construção e documentação na modelagem dos negócios, no processo de desenvolvimento de software e sistemas de informação (FURLAN, 1998) (IBM, 2003).

Assim, a UML surge como uma efetiva ferramenta de modelagem e, a partir da sua extensão com o uso do perfil UML (*UML profile*), apresenta significativo apoio ao projeto físico de banco de dados (NAIBURG & MAKSIMCHURK, 2001) (OMG, 2003) (WHITE et al., 2003).

1.2 OBJETIVO

No geral, o projeto físico de banco de dados é parcialmente realizado durante a engenharia de um banco de dados, restando algumas especificações para quando o banco já estiver implementado e não apresentar, satisfatoriamente, os requisitos de desempenho.

A conversão de uma representação lógica para uma representação física apresenta aparente dificuldade aos projetistas pois, além de requerer elevado grau de conhecimento sobre as características físicas do banco, demanda ainda, um projeto visual sobre as complexas estruturas de armazenamento e recuperação dos dados. Entretanto, alguns aspectos importantes de desempenho dos sistemas de banco de dados, representados no projeto físico, ainda não são suficientemente expressos na UML.

Este trabalho propõe a extensão da UML, mais especificamente o perfil UML de banco de dados, que está em estudo pela OMG (*Object Management Group*), para que aspectos como a indexação e a organização das tabelas do banco, possam ser detalhadas aplicando-se, então, a UML no projeto físico de banco de dados.

Adicionalmente, este trabalho propõe o uso da tabela de freqüência para caso de uso e a especificação de diagrama de seqüência com comandos SQL para as melhores práticas nas tomadas de decisões de um projeto físico de banco de dados.

Alguns dos principais objetivos decorrentes deste trabalho são:

- apresentação das questões críticas num projeto físico de banco de dados;
- apresentação das questões inerentes ao desempenho de um sistema de banco de dados;
- análise dos melhores procedimentos para administração de um banco de dados;
- utilização da UML para modelagem do projeto físico de banco de dados;

- proposição da extensão da UML em projeto físico de banco de dados;
- representação gráfica da modelagem de um projeto físico de banco de dados;
- revisão dos procedimentos adotados atualmente em empresa.

1.3 MOTIVAÇÃO E ESCOPO

O projeto físico de banco de dados é um dos fatores relevantes para a maximização do desempenho de um sistema de informações. Ele direciona a organização na maximização do desempenho do sistema pelo armazenamento planejado dos dados, da recuperabilidade e da segurança, agregando, ainda, facilidades de manutenção do banco de dados. Mas a escolha dos melhores métodos de acesso e recuperação dos dados é uma tarefa complexa, gerando, assim, uma demanda extraordinária para a engenharia de banco de dados.

É dentro deste escopo que o presente trabalho foi elaborado. Conforme ULLMAN (1998), os itens que motivam um projeto de banco de dados são vários, dentre os quais podem ser citados:

- possibilidade de que os usuários tenham facilidade no entendimento do sistema e possam ser colaboradores dos detalhes do projeto;
- atendimento aos requisitos propostos pelos usuários e o provimento de aplicações direcionadas às necessidades do negócio;
- representação detalhada dos níveis lógico e físico para as melhores práticas de armazenamento;
- garantia de que haja desempenho satisfatório para o tempo de resposta no acesso dos usuários ao banco de dados;
- garantia de espaço em disco disponível para os dados atuais e futuros.

Entre as linguagens desenvolvidas para auxiliar os projetistas de sistemas destaca-se a UML. Contudo, a UML ainda não contempla os detalhes do armazenamento físico, como a organização de arquivos e a indexação (GORNIK, 2003). Tal fato motivou este trabalho a buscar no perfil UML os subsídios necessários para apoio ao projeto físico de banco de dados.

Embora a UML seja recente na área de banco de dados, vem crescendo rapidamente, permitindo que seja utilizada para a seleção da estratégia mais apropriada para o armazenamento e recuperação dos dados.

1.4 METODOLOGIA DE TRABALHO

O projeto e a administração de um banco de dados é uma tarefa complexa e analítica por tratar-se de ambiente em constante processo de mudança tecnológica. Deve-se analisá-lo e planejá-lo periodicamente, além da inclusão à busca proativa de novas oportunidades e à solução reativa de problemas existentes na administração do banco de dados (MOLINA, ULLMAN & WIDOM, 2000).

Especificar, detalhadamente, as questões pertinentes ao projeto físico de um banco de dados permitirá que os profissionais da área atuem organizadamente e com maiores critérios, possibilitando a implementação de um banco com melhor desempenho (ULLMAN & WIDOM, 1997).

Partindo desta premissa, foi verificado com o estudo bibliográfico que as dificuldades atuais da maioria dos projetos de banco de dados devem-se à sua complexidade, que exige um elevado grau de conhecimentos específicos, desde a análise de requisitos até a implementação no nível físico. E, notadamente, no nível físico, não ocasionalmente, são mantidas as mesmas práticas viciosas de projeto, com o uso das mesmas organizações de arquivos e índices para as tabelas, criando-se a necessidade de um repensar e representar no projeto os detalhes físicos inerentes ao desempenho do banco de dados. Desta forma, esta pesquisa apoiou-se na UML, considerando esta linguagem de grande importância no aspecto de especificação e modelagem gráfica, sendo possível sua extensão com o uso do *UML Profile*, para que este

trabalho pudesse contribuir de maneira estruturada e consistente com a linguagem UML e, conseqüentemente, com os projetistas no detalhamento dos aspectos físicos no projeto de banco de dados.

1.5 HISTÓRICO TECNOLÓGICO

1.5.1 EVOLUÇÃO TECNOLÓGICA DOS SISTEMAS DE BANCO DE DADOS

No início da era da computação utilizava-se o sistema de arquivos tradicional, ainda hoje utilizado, para armazenamento, manipulação e recuperação dos arquivos de dados. Porém, o sistema de arquivos tradicional apresenta algumas limitações, como (MCFADDEN, HOFFER & PRESCOTT, 1999):

- dependência entre os programas e os dados;
- duplicação dos dados;
- compartilhamento de dados limitado;
- necessidade de programas aplicativos específicos para cada tipo de dados.

No processamento de arquivos tradicional, cada usuário implementa os seus próprios arquivos de acordo com um programa aplicativo específico para manipulá-los, não havendo a centralização desses dados num único local, ocasionando, então, a redundância dos dados. Essa redundância e o armazenamento dos dados resultam na necessidade de espaço físico nos discos e esforços para manter os dados atualizados. Um banco de dados, por outro lado, é um único repositório acessado por vários usuários (DATE, 2001).

Algumas vantagens da utilização do banco de dados sobre o sistema de arquivos podem ser itenizadas da seguinte forma (ELMASRI & NAVATHE, 2000):

- Eliminação das inconsistências para que os mesmos campos tenham os mesmos valores em aplicativos diferentes. Por exemplo, o estado civil de uma pessoa deve ser solteiro num sistema e também no outro sistema, para que não haja inconsistência na informação.

- Padronização dos dados, segundo determinação dos formatos de armazenamento determinados pela empresa. Dessa forma, o campo “Sexo” sempre armazenará o conteúdo “M” ou “F”;

- Redução ou eliminação das redundâncias.

- Os dados que são comuns a vários usuários são compartilhados de modo que uma única informação será consultada por vários sistemas.

- Independência dos dados que representa o armazenamento físico dos dados no banco de dados, sendo que a sua recuperação pelos programas aplicativos é independente do modo como estão armazenados. Assim, quando um programa aplicativo precisa ser alterado, para incluir novos campos, por exemplo, não será necessário alterar a estrutura interna dos arquivos.

- Restrições de segurança de acesso definindo o nível de acesso que cada usuário terá no sistema para o arquivo ou para o campo. Por exemplo, num sistema Workflow pode-se restringir que somente os gerentes poderão aprovar uma ordem de compra.

- Compartilhamento dos dados para que diferentes sistemas e usuários os utilizem de forma segura, consistente e simultânea. A observação é quanto ao gerenciador de concorrentes que deve cuidar para que não ocorram atualizações simultâneas no mesmo campo do mesmo registro.

- Manutenção de integridade para que os campos possuam somente valores válidos. Por exemplo, um usuário não pode colocar na nota fiscal a data de 1500.

Os sistemas de banco de dados tiveram início nos anos 60 e foram evoluindo, década-a-década, buscando superar as limitações anteriores. Alguns importantes acontecimentos contribuíram para o desenvolvimento e a evolução tecnológica dos bancos de dados (MCFADDEN, HOFFER & PRESCOTT, 1999): independência entre programas e dados reduzindo, assim, os custos de manutenção; compartilhamento de dados melhorado; gerenciamento gradativo dos tipos de dados e estruturas; diminuição no risco de ocorrência de

redundância dos dados e fornecimento de acesso mais rápido e fácil aos usuários finais.

1.5.2 REPRESENTAÇÃO DE PROJETOS DE BANCO DE DADOS

Ao longo do tempo, as linguagens de modelagem gráfica vêm sendo utilizadas na engenharia de banco de dados para apoio aos projetistas na representação de banco de dados (KIVISTO, 1999). Dentre várias, destacam-se as linguagens IDEF e a UML, que estão detalhadas no Capítulo 3.

Desenvolvido no início dos anos 70 por Douglas Ross, da Softech, o IDEF0 é conhecido como uma linguagem para Modelagem de Função, sendo derivado da linguagem gráfica SADT (*Structured Analysis and Design Techniques*). Em 1981, a Força Aérea Americana padronizou o ICAM e publicou um subconjunto do SADT, conhecido como IDEF0 (IDEF, 2002) (MARCA & MCGOWAN, 1988).

A Tabela 1 apresenta os 16 modelos IDEF. Cada um é aplicado a um tipo particular de informação criando representações gráficas que auxiliam os projetistas no desenvolvimento dos sistemas.

Tabela 1 - Métodos IDEF
(IDEF, 2002)

MÉTODOS IDEF	
IDEF0	Modelagem da Função
IDEF1	Modelagem da Informação
IDEF1X	Modelagem dos Dados
IDEF2	Modelagem da Simulação do Projeto
IDEF3	Descrição do Processo
IDEF4	Modelagem Orientada ao Objeto
IDEF5	Captura de Descrição de Ontologia
IDEF6	Projeto
IDEF7	Auditando Sistema de Informação
IDEF8	Modelagem da Interface com o Usuário
IDEF9	Modelagem de Cenários
IDEF10	Modelagem da Arquitetura de Implementação
IDEF11	Modelagem da Construção da Informação

IDEF12	Modelagem da Organização
IDEF13	Projeto de Mapeamento do Esquema Triplo
IDEF14	Projeto de Rede

A UML começou a ser desenvolvida no final de 1994 quando Grady Booch e Jim Rumbaugh, da Rational Software, iniciaram a unificação dos métodos Booch e OMT (*Object Modeling Technique*). Em 1995, Ivar Jacobson juntou-se a Rational e assim surgiu o método OOSE (*Object Oriented Software Engineering*). Porém, Booch, Rumbaugh e Jacobson sentiram-se motivados a criar uma linguagem de modelagem mais madura que unificasse a semântica e a notação dos outros métodos já existentes. Em junho de 1996 nascia a versão 0.9 da UML. Em 1997, a OMG (*Object Management Group*) adotou a UML como sua linguagem de modelagem padrão. Desde então, as empresas abraçaram a UML e novas contribuições têm ocorrido (BOOCH, RUMBAUGH & JACOBSON, 1998) (MOK & PAPER, 2001). A Figura 2 ilustra a evolução da UML.

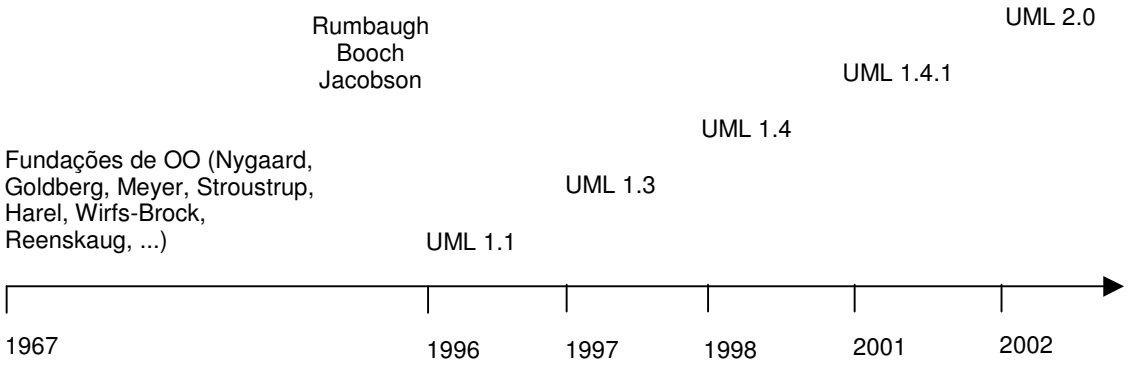

Figura 1 - Evolução da UML
(RATIONAL, 2003)

1.6 ORGANIZAÇÃO DA DISSERTAÇÃO

O presente trabalho está organizado em 6 capítulos sendo que seus respectivos conteúdos estão assim resumidos:

- Introdução (Capítulo 1): apresenta uma visão geral sobre as necessidades que motivaram este trabalho, a metodologia utilizada e os objetivos principais atingidos.

- Engenharia e Administração de Banco de Dados (Capítulo 2): considera os principais requisitos necessários para o projeto de banco de dados e discute o projeto de banco de dados baseado em registros nos níveis: conceitual, lógico e físico, e o projeto orientado a objetos, além de fazer as considerações sobre os modelos de representação IDEF e UML, utilizados amplamente em projetos de sistemas de banco de dados. Por fim, aborda os aspectos mais importantes sobre a administração de banco de dados, notadamente, em sistema relacional.

- Projeto Físico para Banco de Dados (Capítulo 3): discorre sobre os elementos para o projeto físico, especificamente, sobre as estruturas de armazenamento, organização das tabelas e a indexação de arquivos. Apresenta uma representação genérica utilizando a UML como ferramenta gráfica de projeto.

- Extensão do *UML Profile* de Banco de Dados para Projeto Físico (Capítulo 4): os elementos apresentados nos capítulos anteriores são reunidos para propor a extensão da UML em projeto físico de banco de dados e dar a vantagem competitiva aos projetistas e melhores condições visuais do projeto, além de detalhada documentação.

- Estudo de Caso (Capítulo 5): apresenta um estudo de caso onde é aplicada a proposta citada no capítulo anterior.

- Conclusão (Capítulo 6): apresenta a conclusão da dissertação e as contribuições inovadoras que a extensão UML, para projeto físico de banco de dados, pode trazer para a comunidade e as futuras pesquisas que podem ser direcionadas para essa área do conhecimento.

Capítulo 2. ENGENHARIA E ADMINISTRAÇÃO DE BANCO DE DADOS

2.1 CONSIDERAÇÕES INICIAIS

A engenharia de banco de dados tem por finalidade acomodar as informações necessárias aos usuários para um conjunto definido de aplicações e, para tanto, utiliza cinco fases num projeto de banco de dados: requisitos e análises, projeto conceitual, projeto lógico, projeto físico e implementação (ELMASRI & NAVATHE, 2000).

Um banco de dados pequeno geralmente é definido, construído e manipulado por uma pessoa somente. Porém, grandes sistemas necessitam do envolvimento de um número maior de pessoas, dentre as quais destacam-se: (ELMASRI & NAVATHE, 2000):

- Administrador do banco de dados (DBA): administra o sistema de banco de dados, libera ou restringe o acesso aos dados, coordena e monitora o seu uso. Também participa na aquisição de software e hardware necessários para o banco de dados.

- Projetistas do banco de dados: identificam os dados que serão armazenados no banco e escolhem as estruturas mais apropriadas para representar e armazenar esses dados. Entrevistam os potenciais usuários do futuro banco de dados para entender os seus requisitos e elabora um projeto que atenda esses requisitos. Em alguns casos os projetistas são os próprios DBAs ou estão na equipe dos DBAs.

A utilização das ferramentas CASE (*Computer-Aided Software Engineering*) têm colaborado, também, para que a engenharia de banco de dados seja um dos motivos de sucesso na implementação dos sistemas de informação nas empresas (KAMATH et al., 2004).

2.2 REQUISITOS DE BANCO DE DADOS

A finalidade dos requisitos de banco de dados é o estabelecimento de um plano para a coleta das informações pertinentes ao propósito dos sistemas de informação para o negócio. Para o estabelecimento das melhores estratégias de implementação, o projetista deve considerar alguns requisitos importantes para o projeto. O principal objetivo de um projeto físico de banco de dados é o processamento eficiente, para que o tempo de interação dos usuários com o sistema seja o mínimo possível (MCFADDEN, HOFFER & PRESCOTT, 1999).

Avaliar o volume de dados e a freqüência de uso do sistema também são fatores críticos para os requisitos do projeto, embora os esforços devam ser direcionados mais para o processamento eficiente do que para o uso eficiente do espaço em discos (ELMASRI & NAVATHE, 2000).

Alguns fatores de requisitos igualmente importantes são (MCFADDEN, HOFFER & PRESCOTT, 1999):

- Definição da freqüência e do uso dos dados, ou seja, quando e onde serão entrados, recuperados, removidos e atualizados.
- Tamanho e freqüência de uso das tabelas do banco de dados.
- Requisitos para o tempo de resposta esperado pelos usuários do sistema.
- Requisitos para o tempo de backup, recuperação, retenção e integridade.
- Tecnologias de hardware escolhidas e utilizadas para a implementação do banco de dados, incluindo o SGBD.

Num escopo geral, deve ser considerado que os usuários desejam armazenar todo o tipo de dados, porém, para garantir que o projeto atenda os requisitos de desempenho e orçamento planejados, é importante que, preferencialmente, as pessoas que mais conhecem sobre o negócio sejam inquiridas e algumas questões esclarecidas junto aos gestores e projetistas do sistema, sobre a real necessidade dos sistemas de informação para os negócios. As seguintes questões podem ser propostas:

- Quais são os tipos de dados que devem ser armazenados.
- Quantidade de dados envolvidos e manipulados no presente e ao longo do tempo.
- Compartilhamento dos dados entre os usuários.
- Sistemas e tabelas que serão mais acessadas.
- Perspectiva de crescimento anual do banco de dados.
- Quantidade total de usuários do sistema.
- Quantidade de usuários concorrentes do sistema.
- Os dados serão em sua totalidade internos ou também acessados, totalmente ou parcialmente, por parceiros ou outros agentes externos.
- Os dados estarão disponíveis a todos os usuários ou quais grupos de usuários terão acesso aos dados.
- Especificação de níveis de segurança e acesso ao banco para os usuários;
- Os dados dependerão da existência de outros dados e quais dados serão computados a partir de outros dados.
- Quais segmentos do negócio exigem que as saídas de processamento, além de exibidas no monitor de vídeo, também sejam impressas em formulários específicos ou gerais.
- Centralização ou distribuição dos equipamentos de armazenamento de dados e servidores corporativos.
- Integração com outros sistemas ou banco de dados já existentes.
- Especificação da rede de dados para suportar o novo cenário.
- Especificação de servidores de banco de dados e aplicativos.

- Especificação mínima de hardware para os usuários para que não ocorram problemas de desempenho por equipamentos inadequados.

- Especificação de equipamentos e software de backup para o banco de dados.

2.3 ENGENHARIA DE BANCO DE DADOS RELACIONAL

O modelo relacional (CODD, 1990) foi introduzido em 1970 por E. F. Codd e utiliza um conjunto de tabelas para representar os dados e os seus relacionamentos. Tem sido um dos modelos comerciais mais utilizados devido a sua simplicidade e fundações matemáticas com tabelas bidimensionais. Logo, pode-se afirmar que um banco de dados relacional é uma coleção de tabelas bidimensionais. Cada tabela é formada por várias colunas (ou atributos) e cada coluna possui um nome único. As colunas das tabelas contém informações sobre a tabela e as linhas (ou tuplas) da tabela representam as ocorrências representadas pela tabela. Um dado é armazenado na intersecção de uma linha e uma coluna. Cada coluna possui um domínio que é um conjunto de valores que podem aparecer naquela coluna (ELMASRI & NAVATHE, 2000). A Figura 2 ilustra um exemplo de um banco de dados relacional apresentado em duas tabelas:

nome_cliente	**seguro_social**	**rua_cliente**	**cidade_cliente**	**número_conta**
Johnson	192-83-7465	Alma	Palo Alto	A-101
Smith	019-28-3746	North	Rye	A-215
Hayes	677-89-9011	Main	Harrison	A-102
Turner	182-73-6091	Putnam	Stamford	A-305
Johnson	192-83-7465	Alma	Palo Alto	A-201
Jones	321-12-3123	Main	Harrison	A-217
Lindsay	336-66-9999	Park	Pittsfield	A-222

número_conta	**saldo**
A-101	500
A-215	700
A-102	400
A-305	350
A-201	900
A-217	750
A-222	700

Figura 2 - Banco de Dados Relacional
(SILBERSCHATZ, KORTH & SUDARSHAN, 2002)

Um banco de dados fornece aos usuários uma visão abstrata dos dados para que haja uma simplificação na interação deles com o sistema. Porém, em decorrência da complexidade inerente do banco de dados, alguns detalhes de como os dados são armazenados não são do conhecimento dos usuários finais ficando restrito aos projetistas do banco. Assim, o planejamento adequado das melhores condições de implementação do sistema é feito através de algumas fases da engenharia de banco de dados, que abrangem desde o nível mais alto da estrutura do banco, representando a visão dos usuários sobre o banco de dados, até o nível mais baixo, que representa quais dados e o modo como esses dados serão armazenados e recuperados no banco de dados (SILBERSCHATZ, KORTH & SUDARSHAN, 2002). A Figura 3 apresenta as fases da engenharia de banco de dados para o modelo relacional:

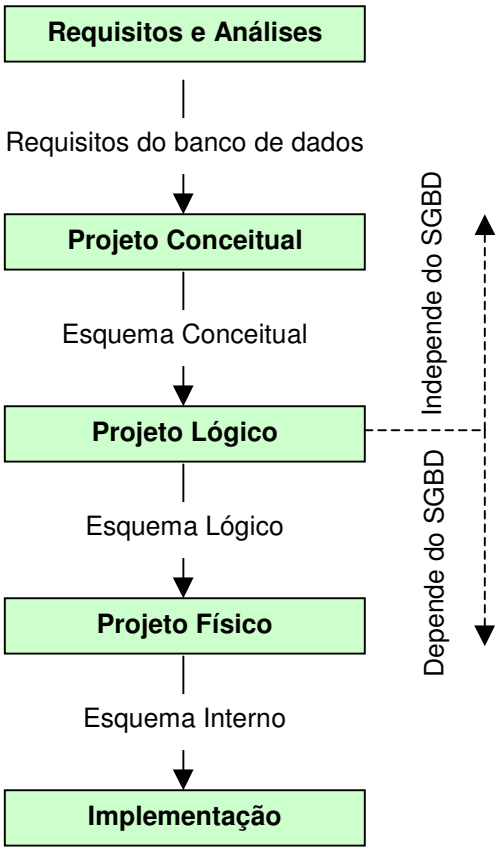

Figura 3 - Fases da Engenharia de Banco de Dados
(ELMASRI & NAVATHE, 2000)

2.3.1 REQUISITOS E ANÁLISES

Uma engenharia de banco de dados inicia-se com a coleta e a análise dos requisitos de dados dos potenciais usuários do banco por meio de entrevistas e outros meios, objetivando entender quais são as suas necessidades quanto aos dados (SILBERSCHATZ, KORTH & SUDARSHAN, 2002).

Os projetistas conhecem bem as técnicas para o desenvolvimento do projeto do banco de dados mas, de um modo geral, podem não entender do negócio; logo, precisam levantar com os usuários quais são os dados e como esses dados serão armazenados no banco, bem como quais serão os programas aplicativos necessários a cada área específica do negócio. Essas entrevistas podem acontecer várias vezes até que as informações fornecidas pelos usuários estejam coerentes, sem ambigüidades, contradições ou qualquer mudança mais significativa.

As seguintes atividades que contemplam essa fase podem ser assim resumidas (ELMASRI & NAVATHE, 2000):

- Identificação dos usuários ou grupos de usuários principais e das maiores áreas da aplicação do banco de dados.

- Entrevistas com os usuários levantando as prioridades e a importância das aplicações para eles.

- Estudos e análises das documentações referentes as aplicações, conforme as entrevistas e informações coletadas junto aos usuários e observações constatadas no conhecimento do negócio.

- Estudo do ambiente operacional da empresa e o objetivo planejado do uso da informação para atendimento aos requisitos atuais e futuros.

No início existirão muitas informações incoerentes, imprecisas, incorretas e que precisarão de novas entrevistas até que esse processo repetitivo seja cada vez mais refinado e as informações transformadas em requisitos que reflitam o entendimento inicial do sistema (KELLNER & ROMBACH, 1991).

A fase de requisitos e análises pode ser demorada conforme a complexidade do negócio. Porém, será mais oneroso remediar um erro no final do projeto do que no início.

2.3.2 PROJETO CONCEITUAL

O projeto conceitual é iniciado após a coleta e a análise dos requisitos e através de um modelo de dados conceitual de alto nível, como o Modelo Entidade-Relacionamento (BAGUI & EARP, 2003), cria-se um esquema conceitual para o banco de dados independentemente do SGBD (ELMASRI & NAVATHE, 2000). Esse esquema assegurará que os requisitos dos usuários estarão sendo atendidos, pois apresenta uma visão de alto nível da estrutura do banco de dados, ou seja, o modo como os usuários finais visualizarão as partes ou todos os dados do banco de dados. Geralmente é um esquema de fácil entendimento por pessoas leigas, além de não abordar detalhes do armazenamento físico dos dados podendo, assim, ser discutido pelos colaboradores dos requisitos e gestores do negócio. Quanto maior o nível de detalhes do esquema conceitual, melhor será o projeto lógico (ULLMAN, 1998).

Nessa fase serão abordadas, ainda, as especificações das necessidades funcionais da empresa e quais os tipos de operações ou transações que os usuários realizarão no banco de dados. Por meio desses estudos planeja-se, ainda, os programas aplicativos que melhor atendem a empresa na manipulação desses dados (SILBERSCHATZ, KORTH & SUDARSHAN, 2002).

O esquema conceitual descreve os requisitos dos usuários incluindo os tipos das entidades, as restrições e os relacionamentos. Essa fase do projeto deve ter as seguintes características (ELMASRI & NAVATHE, 2000):

- Expressividade para distinguir os diferentes tipos de dados, relacionamentos e restrições.

- Simplicidade para que os usuários não especialistas entendam os conceitos do projeto e possam participar ativamente com os detalhes das informações.

- Número mínimo de conceitos com significados ambíguos.

- Representação diagramática para fácil interpretação.

- Formalidade para exprimir exatamente o projeto.

A Figura 4 ilustra um exemplo genérico de um Modelo Entidade-Relacionamento correspondente a clientes e empréstimos.

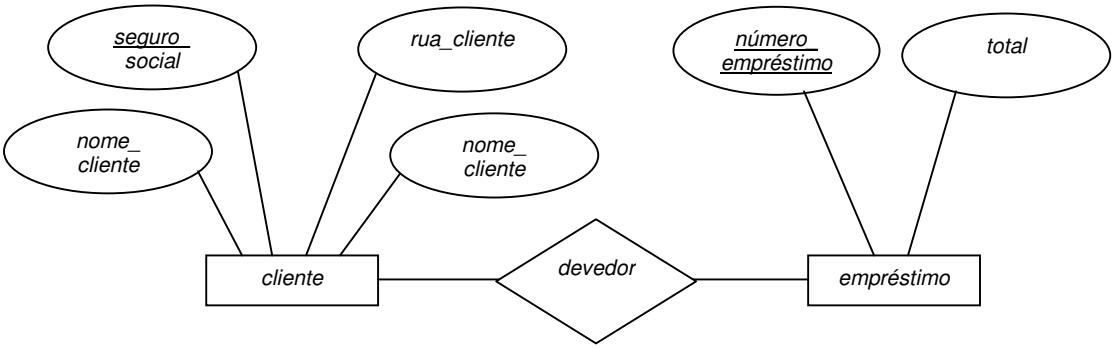

Figura 4 - Diagrama Entidade-Relacionamento
(SILBERSCHATZ, KORTH & SUDARSHAN, 2002)

Alguns elementos gráficos do Modelo Entidade-Relacionamento, que aparecem na Figura 4, são detalhados a seguir:

- Retângulos: conjuntos de entidades

- Elipses: atributos

- Losangos: conjuntos de relacionamentos

- Linhas: união dos atributos aos conjuntos de entidades e os conjuntos de entidades aos conjuntos de relacionamentos

- Elipses duplas: atributos multivalorados

- Linhas duplas: participação total de uma entidade em um conjunto de relacionamentos

2.3.3 PROJETO LÓGICO

O projeto lógico é iniciado a partir do esquema conceitual e o objetivo é a implementação do modelo de dados conceitual em um modelo de dados lógico, conforme um SGBD específico. O resultado dessa fase serão as sentenças

DDL (*Data Definition Language*) conforme o SGBD escolhido (MCFADDEN, HOFFER & PRESCOTT, 1999). Esse processo geralmente é feito com o auxílio de ferramentas CASE ou com a utilização da UML (LARMAN, 2000). O modelo de dados relacional representa os dados na forma de tabelas, chamadas de relações ou tabelas bidimensionais de dados.

Apesar dos projetistas já saberem quais são os dados que serão armazenados e terem definidos os seus atributos, o projeto ainda poderá sofrer alterações nessa fase, se necessário. Seja em função de alguma mudança operacional ou estratégica definida pela empresa (MCFADDEN, HOFFER & PRESCOTT, 1999).

Para exemplificar, a Figura 5, ilustra o diagrama de projeto lógico Integração_GL, parte do produto AP - *Oracle Payables* (Folha de Pagamento), do aplicativo Oracle versão 11.5.9 (11i), da empresa Oracle Corporation.

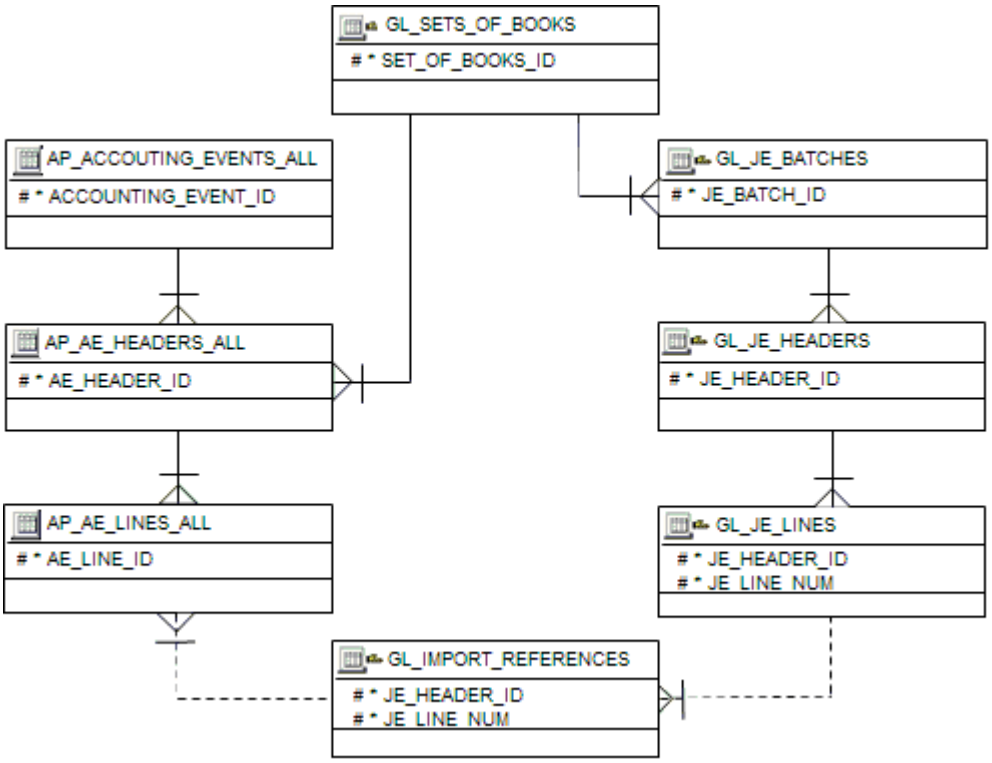

Figura 5 - Diagrama Integração_GL
(ORACLE, 2004)

No projeto lógico são também apresentados mais detalhes da implementação dos dados, visando atender as metas de desempenho de acesso e as

facilidades para a manutenção do banco. Apresenta, na sua fase final, as especificações dos dados transformados em elementos básicos ou atômicos, seguindo regras bem estabelecidas e especificações de dados bem estruturadas que, regularmente, seguem a teoria do banco de dados relacional, num processo chamado normalização (ELMASRI & NAVATHE, 2000).

Mesmo um bom projeto de banco de dados pode apresentar estruturas de tabelas ruins e a normalização é um processo para avaliação e correção das estruturas de tabelas e decomposição das relações anômalas com o objetivo de se produzir relacionamentos menores e bem estruturados reduzindo, assim, as redundâncias dos dados (MCFADDEN, HOFFER & PRESCOTT, 1999).

A normalização é amplamente utilizada em projetos de bancos relacionais, pois valida e melhora o projeto lógico, prevenindo a duplicidade desnecessária dos dados, minimizando o espaço utilizado pelos dados no dispositivo de armazenamento, além de garantir a integridade e confiabilidade da informação. Logo, a normalização e a modelagem Entidade Relacionamento são normalmente utilizadas conjuntamente (MCFADDEN, HOFFER & PRESCOTT, 1999).

2.3.4 PROJETO FÍSICO

O projeto físico de um banco de dados é um processo que envolve tarefas difíceis e que consomem muito tempo. Dentre essas tarefas destacam-se as especificações das estruturas de armazenamento e organização e métodos de acesso aos arquivos do banco de dados (CHOENNI, BLANKEN & CHANG, 1993).

O projeto físico deve acomodar a estimativa de um alto volume de dados e um tempo de resposta rápido para suportar um alto volume de transações. Isso inclui as estruturas de armazenamento e os índices que otimizam a pesquisa e a recuperação dos dados. Durante essa fase deve-se planejar as melhores estratégias para obter acesso rápido aos registros em um arquivo no disco usando uma estrutura de índice adequada (ELMASRI & NAVATHE, 2000).

Os índices aumentam a velocidade de recuperação dos registros baseados em algumas condições de pesquisa podendo acessá-los sem a necessidade de reposicioná-los em outro local do disco. A maioria dos tipos de índices é baseada em arquivos ordenados e estruturas de dados do tipo árvore (MOLINA, ULLMAN & WIDON, 2001).

Detalhes gerais sobre o projeto físico serão apresentados no Capítulo 3.

2.3.5 IMPLEMENTAÇÃO

A última fase da engenharia de banco de dados é a implementação em dispositivo de armazenamento maciço de dados e, no caso da maioria dos modelos relacionais, isso é feito por meio de uma linguagem de pesquisa estruturada de comunicação com o banco de dados, conhecida como SQL - *Structured Query Language* (YAO, 1979). A versão SQL mais comumente utilizada é o SQL-92 que, inclusive, foi a base para o JBDC, SQLJ e SQL:2003, sendo essa última a versão mais recente da linguagem. À medida que os bancos de dados forem migrados para a SQL:2003 (ISO/IEC 9075), se tornarão propícios para atender ao paradigma relacional-objeto.

Os quatro maiores grupos da linguagem SQL são (MELTON & SIMON, 2002):

- Linguagem de Definição de Dados (DDL).
- Linguagem de Manipulação de Dados (DML).
- Linguagem de Pesquisa de Dados (DQL).
- Linguagem de Controle de Dados (DCL).

A Linguagem de Definição de Dados é um conjunto de comandos utilizados para criar, modificar e remover as estruturas do banco de dados, além de controlar o acesso aos objetos do banco. Geralmente é utilizado por um administrador do banco de dados, projetista de banco ou mesmo um desenvolvedor de aplicativos. Entre os comandos DDL estão o *drop*, *truncate*, *create* e *alter*. Exemplo de um comando DDL para a criação de uma tabela chamada pessoa:

```
create table pessoa
    (nome           VARCHAR2(50),
    endereco        VARCHAR2(70),
    numero          VARCHAR2(10),
    cep             VARCHAR2(9),
    cidade          VARCHAR2(40),
    uf              VARCHAR2(2))
```

A Linguagem de Manipulação de Dados é um conjunto de comandos utilizados para a manipulação dos dados dentro do banco de dados. Os comandos DML são: *insert*, *delete* e *update*. Exemplo da remoção de uma linha da tabela pessoa:

```
delete from apps.pessoa
    where nome = 'joão da silva'
```

A Linguagem de Pesquisa de Dados é utilizada para obter os dados do banco. O comando DQL é o *select*. Exemplo de um *select* na tabela pessoa:

```
select * from apps.pessoa
```

A Linguagem de Controle de Dados é utilizada para controlar os privilégios de acessos aos dados. Entre os comandos da DCL estão o *revoke* e *grant*. Exemplo do comando *grant* para garantir inserção na tabela pessoa para o usuário1:

```
grant insert on apps.pessoa to usuário1
```

2.4 MODELOS DE REPRESENTAÇÃO

2.4.1 METÁFORAS DA REPRESENTAÇÃO

O crescente número de usuários que fazem uso dos sistemas de computadores e a crescente demanda por funcionalidades, tem tornado os sistemas incrivelmente mais complexos, além das dificuldades já inerentes na engenharia de banco de dados. Ainda, a parte física que suporta um sistema também está mais complexa. Os processadores estão mais elaborados, há

maiores sistemas de memórias, os sistemas operacionais apresentam mais funcionalidades e os meios de transmissão estão mais rápidos (BOSCH et al., 2000).

A metáfora[1] tem sido utilizada para melhorar a interação do homem com a máquina. Os arquivos e diretórios são criados, movidos, removidos e copiados como se fossem realmente objetos físicos. A metáfora possui essa capacidade de transformar o abstrato, de fazer um novo sistema parecer um sistema já conhecido, facilitando o entendimento e, efetivamente, contribuindo para um sistema melhor. No projeto de um sistema, o projeto conceitual é feito a partir de um grupo de usuários sobre um contexto particular. Diante disso, esboçam uma comunicação visual sobre a estrutura e navegação do sistema e como essas partes estarão interligadas, conforme a Figura 6 (MOHNKERN, 1997).

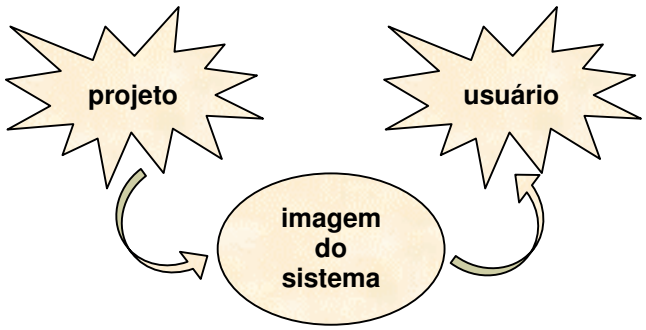

Figura 6 - Comunicação Visual
(MOHNKERN, 1997)

Assim, as metáforas são conceitos fundamentais, termos e imagens pelas quais e por meio das quais a informação é facilmente reconhecida, entendida, documentada e relembrada (MARCUS, 1997).

As etapas básicas no processo de metáforas da representação incluem a modelagem e o gerenciamento dos dados, fornecendo representações visuais dos dados e o mapeamento dos dados para o visual. Isso também inclui as condições necessárias para que os usuários interajam com os dados (BOSCH et al., 2000) (LEE & GRINSTEIN, 1995).

---

[1] "*Emprego de uma palavra em sentido diferente do próprio por analogia ou semelhança"* (Dicionário Prático da Língua Portuguesa, 1995). A metáfora representa a comparação subjetiva entre duas coisas, como no caso de um desenho de estrutura residencial, no qual são expressas as idéias do engenheiro para visualização e análise da concepção do mesmo.

Os sistemas de informações eram como um componente de suporte aos negócios, sendo que tinham uma atuação mais efetiva na qualidade e funcionalidade dos serviços. Porém, cada vez mais eles tornam-se componentes dos negócios, de modo que hoje o próprio negócio define os requisitos para os seus sistemas de informações (CURTIS, KELLNER & OVER, 1992). Não raramente, os sistemas conseguem suportar corretamente os negócios, seja pela falta de definição e clareza dos requisitos, seja pela carência no entendimento do negócio pela equipe do projeto ou ainda pela própria natureza do negócio, que muda tão rapidamente que o sistema não consegue acompanhá-lo. Os sistemas de informações atuais têm vários problemas de inflexibilidade, complexidade, suporte ineficiente aos negócios e documentações textuais e visuais precárias (NORAN, 2004).

As ferramentas de visualização gráfica são um efetivo instrumento para auxiliar o entendimento do banco de dados através de representações gráficas que reduzem a complexidade, melhoram a documentação, além de serem tão importantes quanto as documentações textuais (MACKINNON & MURPHY, 2003) (TILLEY & HUANG, 2003).

Logo, as ferramentas de modelagem, como o IDEF (*Integrated DEFinition*) e a UML (*Unified Modeling Language*), ganham, cada vez mais, adeptos nos meios acadêmicos e corporativos pelas facilidades com que representam os sistemas e os aspectos pertinentes as áreas de negócios das empresas. A Figura 7 ilustra uma representação genérica de um modelo de negócios e sistemas de informações.

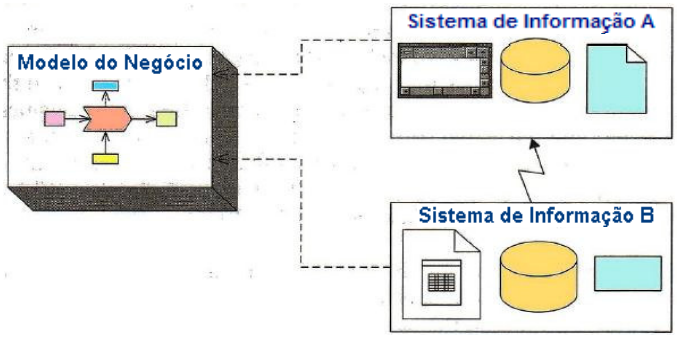

Figura 7 - Modelo de Negócios e Sistemas de Informações
(NORAN, 2004)

2.4.2 MÉTODOS DE DEFINIÇÃO INTEGRADOS (IDEF)

O IDEF - *ICAM DEFinition* e, a seguir, *Integrated DEFinition* foi, inicialmente, desenvolvido para ser aplicado em engenharia de sistemas, mas, a necessidade crescente dos programas de projetos e análises suportarem também metodologias de modelagem, fez com que o programa ICAM (*Integrated Computer-Aided Manufacturing*), da Força Aérea Americana, fosse planejado com um conjunto de métodos conhecido como IDEF (CHO, 2000). Assim, o IDEF passou a ser um conjunto de métodos que podem ser utilizados para a modelagem de vários sistemas, incluindo as operações e áreas de negócios das empresas, criando uma abstração complexa do ambiente, assegurando aos gestores as condições necessárias para que as funções do negócio possam ser melhor entendidas e, por conseguinte, possam então promover as alterações e inovações necessárias (IDEF, 2002).

IDEF0

As aplicações iniciais do IDEF0 foram direcionadas para auxiliar na melhoria da produtividade de manufatura pela aplicação de métodos estruturados para que os processos fossem diagramados pelos seus executores, da mesma maneira com que eram conhecidos e realizados (IDEF, 2002).

O IDEF0 tem sido utilizado para modelagem das atividades ou perspectivas funcionais de um sistema, permitindo que os usuários descrevam os processos através de setas que entram e saem numa caixa, representando alguns aspectos como: Entradas, Saídas, Controles e Mecanismos. A Figura 8 ilustra um metamodelo genérico do IDEF0 (NORAN, 2004).

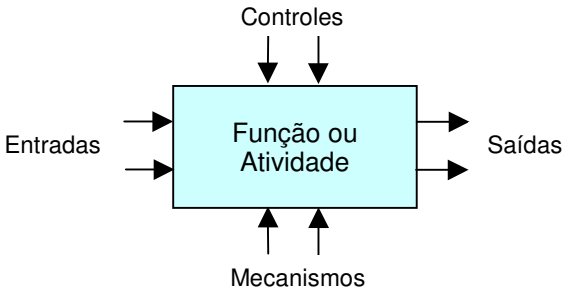

Figura 8 - Diagrama Genérico IDEF0
(NORAN, 2004)

Os aspectos que aparecem na Figura 8 são detalhados a seguir:

- Entradas: informações ou recursos necessários para a realização da atividade (processo) que serão consumidos ou transformados.
- Saídas: resultados ou coisas geradas pela transformação ou consumo da atividade.
- Controles: controlam por que ou como uma atividade é realizada através de políticas, regras, leis, etc.
- Mecanismos: agentes que completam as atividades contidas no processo, como: pessoas ou ferramentas manuais ou automáticas.

É importante ressaltar, ainda, que o diagrama IDEF0 não representa uma seqüência ou fluxo, mas apenas as atividades, por isso, geralmente é utilizado em conjunto com outros métodos IDEF, como o IDEF3. Pode ser, contudo, decomposto em vários diagramas de níveis inferiores ordenadamente numerados em diagramas pais e filhos. A Figura 9 ilustra um exemplo de um diagrama IDEF0 para um sistema de compra. Para representar um processo completo os blocos de Função ou Atividade são relacionados uns com os outros. Assim, a ordem de compra, que é uma saída no primeiro bloco, torna-se tanto uma entrada quanto um controle no segundo bloco (NORAN, 2004).

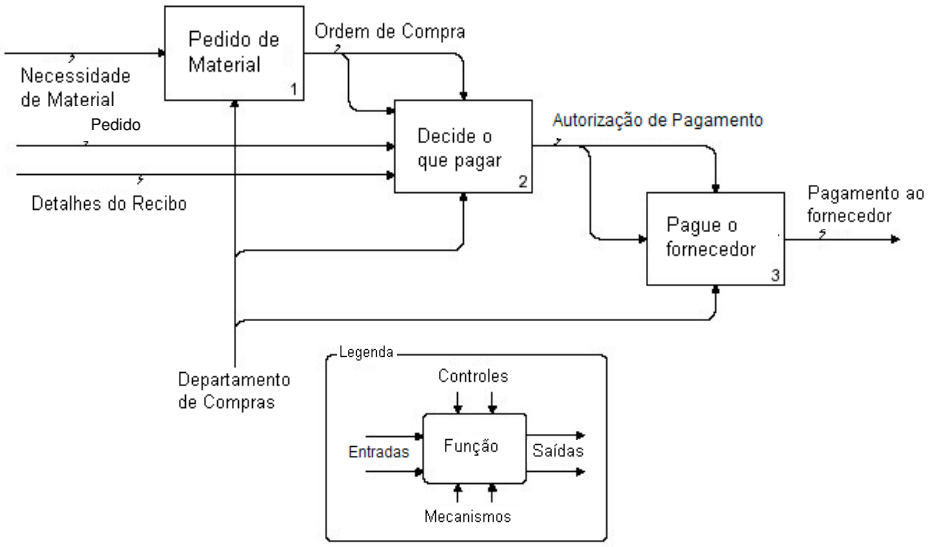

Figura 9 - Diagrama IDEF0
(NORAN, 2004)

IDEF1

O IDEF1 é um método que foi derivado do Atributo_Entidade (*Entity_Link_Attribute*), Entidade Relacionamento (ER) e Modelo Relacional de Codd; sua principal aplicação é a análise e a comunicação no estabelecimento dos requisitos informacionais e não informacionais como pessoas, telefones, etc, ou seja, é um método para o levantamento das informações dos objetos conceituais ou físicos existentes na empresa para a modelagem da informação (IDEF, 2002).

O IDEF1 não é um método para projeto físico de banco de dados, mas um meio que poderá emergir alguns aspectos importantes dos sistemas para que os administradores dos sistemas de informação busquem um melhor planejamento estratégico e atinjam a vantagem competitiva da empresa, melhorando a política e o gerenciamento do ambiente informacional. Esses aspectos são os seguintes (NORAN, 2004):

1) informação coletada, armazenada e gerenciada pela empresa;

2) regras que conduzem o gerenciamento da informação;

3) relacionamentos lógicos refletidos na informação;

4) problemas resultantes da falta de um bom gerenciamento da informação.

Utilizando representações gráficas simples o IDEF1 colabora no desenvolvimento de um modelo por meio de um processo disciplinado e estruturado para a análise e o gerenciamento da informação. No entanto, há alguns conceitos importantes que devem ser considerados no IDEF1 para a determinação dos requisitos da informação. Os principais conceitos envolvidos no método IDEF1 são (IDEF, 2002):

- Entidades: representam a informação sobre os objetos conceituais e físicos da empresa.

- Atributos: são as características das entidades.

- Relações: representam o relacionamento entre as entidades.
- Classes: são os conjuntos de entidades ou atributos.

A Figura 10 mostra um exemplo de uma relação no IDEF1 na qual há uma ligação entre as entidades *Department* e *Employee*. Os diamantes no final ou no meio da ligação informam a cardinalidade e dependência.

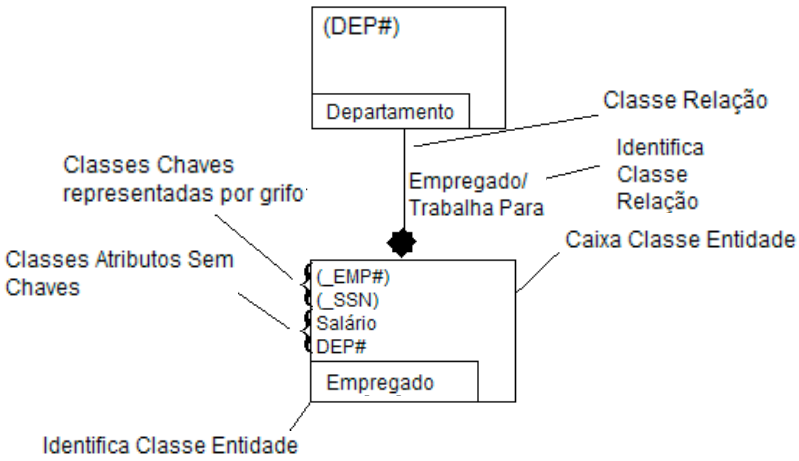

Figura 10 - Diagrama IDEF1
(IDEF, 2002)

IDEF1X

O IDEF1X tem uma terminologia muito similar ao IDEF1, mas há diferenças importantes entre eles. Enquanto o IDEF1 é aplicado à modelagem da informação, o IDEF1X é um método para a modelagem dos dados. Ele é baseado no Modelo Entidade Relacionamento de Chen (CHEN, 1976) para o propósito de projeto lógico de banco de dados relacional, mas como não segue as regras de um bom projeto gráfico – seus símbolos não são totalmente evidentes, também não é um método adequado para o projeto físico de banco de dados relacional ou orientado a objetos (IDEF, 2002).

Os conceitos principais do IDEF1X são: entidade (referente a uma coleção), atributo e estrutura de classificação para modelagem dos tipos de dados lógicos.

O IDEF1X requer a especificação de uma classe chave para distinguir uma entidade da outra, porém, os sistemas orientados a objetos não requerem

chaves para individualizar um objeto do outro, sendo aplicado em projetos nos quais já foram reconhecidos os requisitos da informação e tomada a decisão pelo uso do modelo relacional. A Figura 11 ilustra um exemplo de diagrama IDEF1X.

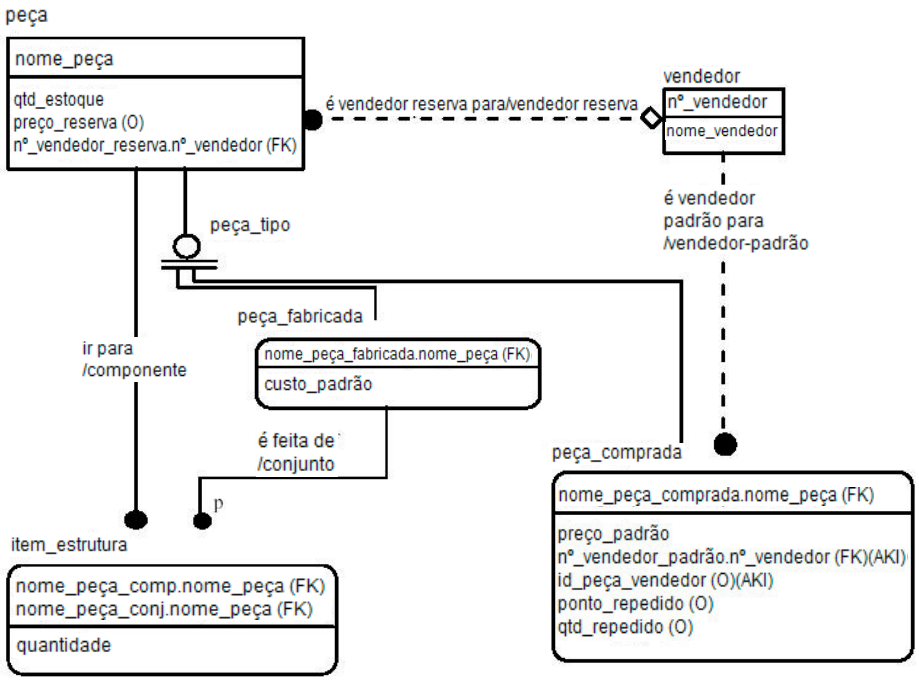

Figura 11 - Diagrama IDEF1X
(IDEF, 2002)

IDEF2 e IDEF3

O IDEF2 trata do modelo da simulação para que possam ser analisados os aspectos mais complexos e seus efeitos no sistema proposto ou atual (NORAN, 2004).

O IDEF3 é um método para a coleta e documentação dos processos atuais e futuros, expressando conhecimento sobre como um sistema, processo ou empresa funciona (IDEF, 2002).

Embora similar ao IDEF0, o IDEF3 possibilita que as visões dos usuários sobre os processos sejam decompostas em várias descrições, enquanto que o IDEF0 engloba somente uma única visão do sistema.

O IDEF3 descreve os aspectos comportamentais (dinâmicos) de um sistema e utiliza dois maiores métodos:

- DFP (Descrição do Fluxo de Processo) ou PFD (Process Flow Description);
- RTEO (Rede de Transição Estado-Objeto) ou OSTN (Object State Transition Network).

O DFP descreve o modo como parte de um sistema ou processo funciona seqüencialmente num fluxo de processo no cenário em que está inserido. Abaixo segue um exemplo genérico do DFP. O elemento básico IDEF3 representado pelas caixas é o UOB (*Unit Of Behavior*) e cada caixa pode descrever atividades, processos e eventos e ser decomposta em outras UOB. O fluxo do processo é verificado através das setas que interligam as caixas conforme o exemplo da Figura 12 de um diagrama de fluxo de processo (NORAN, 2004).

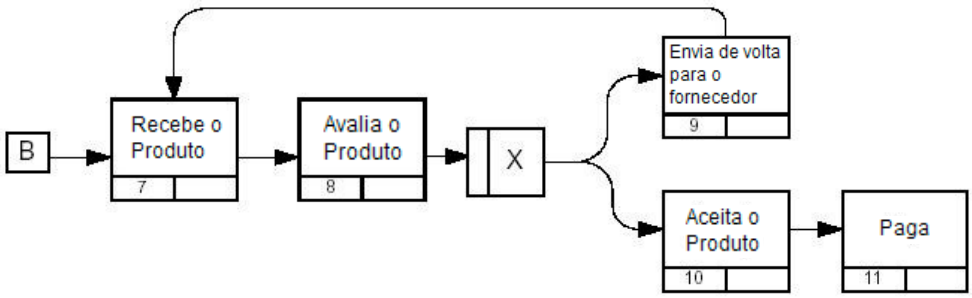

Figura 12 - Diagrama IDEF3 de Fluxo de Processo
(NORAN, 2004)

Já o diagrama RTEO representa uma visão centrada-objeto dos processos, ou seja, resume as transições permitidas por meio de cortes nos diagramas de processos. Nos diagramas RTEO os círculos representam os estados-objetos e as setas que conectam os círculos são os estados de transição. A definição de um estado-objeto está relacionada à condição continuada do objeto naquele estado. As condições de entrada e saída caracterizam os requisitos antes que o objeto transite dentro de um estado e as condições para que ele possa transitar fora de um estado. A Figura 13 ilustra um exemplo do RTEO (NORAN, 2004).

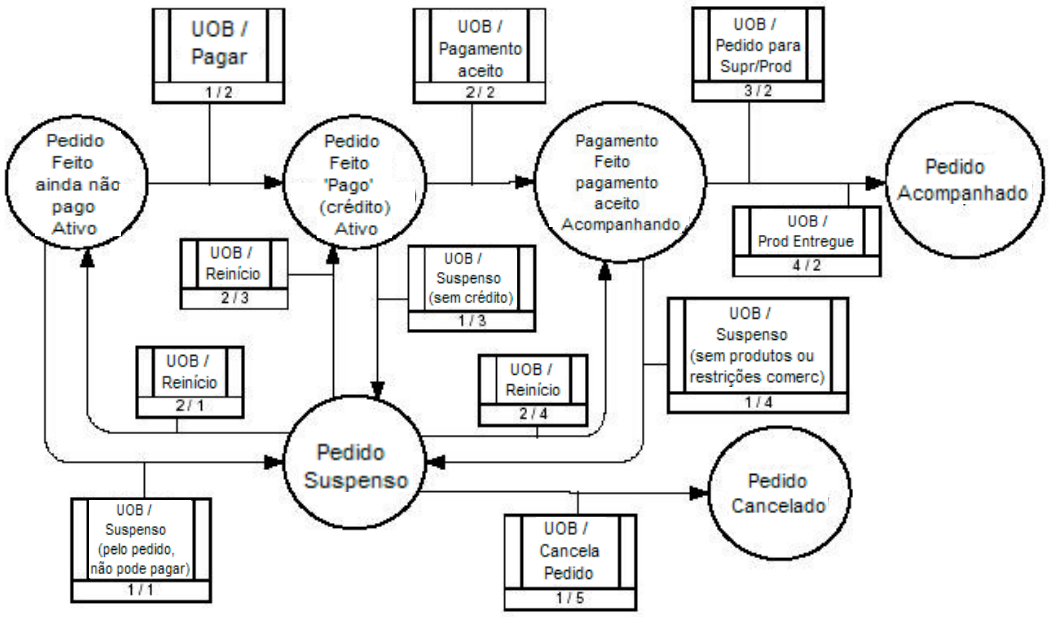

Figura 13 - IDEF3 de Rede de Transição Estado-Objeto
(NORAN, 2004)

2.4.3 LINGUAGEM DE MODELAGEM UNIFICADA (UML)

A UML é uma linguagem para especificação, visualização, construção e documentação para modelagem dos negócios e no processo de desenvolvimento de software, pois, por meio de símbolos, conhecidos como notações, e regras, conhecidas como semânticas, pode representar tanto os aspectos conceituais quanto lógicos (IEEE, 2003) (OMG, 2003). Mas ainda há muita discussão sobre a sua aplicabilidade como ferramenta de modelagem para projeto físico de banco de dados porque, embora ela possa representar graficamente as estruturas físicas através dos seus componentes estereotipados, ainda carecem detalhes sobre outros aspectos igualmente importantes do projeto físico de banco de dados, como: indexação e organização dos arquivos, tipos de índices e tamanhos dos blocos de dados (BJÖRKANDER & KOBRYN 2003) (DORSEY, 2003) (RATIONAL, 2003).

A UML contém um conjunto de notações e regras que gerenciam a linguagem. As regras podem ser classificadas como (FOWLER, 2004):

- Sintática: especifica o aspecto e a combinação das regras.

- Semântica: especifica o significado dos símbolos, individualmente e no contexto;

- Pragmática: especifica as linhas gerais sobre como utilizar a linguagem.

Há dois conceitos importantes sobre a UML: diagramas e mecanismos de extensão (WHITE et al., 2003).

Diagramas da UML

Há muitos diagramas da UML utilizados para a modelagem dos negócios ou sistemas. A Figura 14 apresenta os principais tipos.

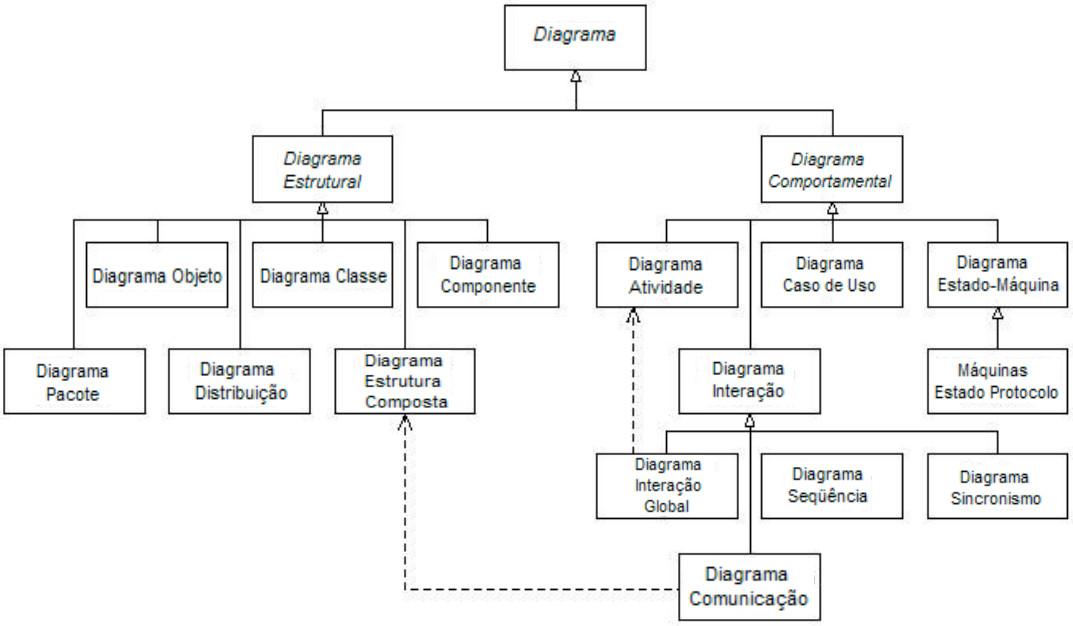

Figura 14 - Diagramas da UML
(RATIONAL, 2003)

Um diagrama pode ser categorizado em diagrama estrutural, comportamental ou de interação. Um diagrama estrutural é utilizado para a construção dos blocos do sistema, ou seja, as características que não mudam durante o tempo. O diagrama comportamental representa como o seu sistema responderá as solicitações, ou seja, ele evoluirá ao longo do tempo e o diagrama de interação é um tipo de diagrama comportamental que descreve

um grupo de objetos que se relacionam (FOWLER, 2004). A Tabela 2 descreve alguns tipos de diagramas da UML e suas utilizações (OMG, 2003).

Tabela 2 - Descrição dos Diagramas da UML
(OMG, 2003)

Categorias	Tipos de Diagramas	Propósitos
Estrutural	Diagrama Classe	Usado para exibir as entidades do mundo real, elementos de análise, projeto e seus relacionamentos.
Estrutural	Diagrama Objeto	Especifica ou ilustra exemplo de objetos e seus relacionamentos.
Estrutural	Diagrama Estrutura Composta	Representa como alguma coisa é realizada. Muito utilizado em projeto baseado em componente.
Estrutural	Diagrama Distribuição	Representa a arquitetura do sistema como: hardware e software.
Estrutural	Diagrama Componente	Utilizado para representar a organização e os sistemas distribuídos.
Estrutural	Diagrama Pacote	Organiza os modelos e representa a relação entre eles.
Comportamental	Diagrama Atividade	Usado para representar o fluxo de dados.
Comportamental	Diagrama Caso de Uso	Representa o que os atores podem solicitar do sistema.
Comportamental	Diagrama de Estado	Mostra o ciclo de vida de um objeto em particular ou as seqüências de um objeto ao longo de sua vida.
Interação	Diagrama Geral	Usado para mostrar muitos diferentes cenários de interações (seqüências de comportamentos) para a mesma colaboração (conjunto de elementos trabalhando juntos).
Interação	Diagrama Seqüência	Representa a ordem das mensagens entre um grupo de objetos.
Interação	Diagrama Comunicação	Relacionamento subjacente entre os objetos.

Devido à flexibilidade da UML os diagramas podem ser categorizados como:

- Diagramas Estáticos: Mostram as características estáticas do sistema. São muito similares aos diagramas estruturais.
- Diagramas Dinâmicos: Representam a evolução do sistema ao longo do tempo.
- Diagramas Funcionais: Exibem os detalhes do comportamento do sistema diante das solicitações realizadas.

2.5 PERFIL UML PARA PROJETO FÍSICO DE BANCO DE DADOS

A UML descreve as propriedades para cada um dos elementos modelo e possibilita mecanismos de extensão para que novos modelos sejam criados. O perfil constitui o principal mecanismo que a UML oferece para estender a sua sintaxe e semântica, adaptando os modelos UML a um domínio específico de aplicação (ALHIR, 1998) (SELONEM, KOSKIMIES & SAKKINEN, 2001).

O perfil se define com base em três mecanismos de extensão da UML: estereótipos, descrições das classes e regras. Estereótipo é um elemento de modelagem UML que estende o metamodelo para aplicações em domínios específicos. A descrição da classe permite a inclusão de informação sobre um elemento padrão UML e a regra limita o comportamento do elemento UML (CONEJERO, HERNÁNDEZ & PEDRERO, 2006) (TERRASSE, SAVONNET & BECKER, 2001).

Pela proposta do OMG de definição de *profiles* para modelagem de aspectos específicos de sistema, estão sendo pesquisados, definidos e padronizados *profiles* que ajudam os projetistas de sistema. Alguns *profiles* já tornaram-se padrões (UML, 2005) como o *Human-Usable Textual Notation* (HUTN), *UML Profile for CORBA*, *UML Profile for CORBA Component Model* (CCM), *UML Profile for Enterprise Application Integration* (EAI), UML Profile for Enterprise Distributed Object Computing (EDOC), *UML Profile for QoS and Fault Tolerance*, *UML Profile for Schedulability*, *Performance and Time* e *UML Testing Profile.* No caso de banco de dados, diversas pesquisas estão sendo realizadas para sua definição, sendo que esse trabalho de pesquisa foca o

mesmo objetivo (AKEHURST et al., 2002) (MORA & TRUJILLO, 2004) (NASSAR, 2003).

O Perfil UML para Banco de Dados, atualmente em estudo pela OMG, é apresentado como uma extensão na UML com o uso de metáforas de representação para suporte a modelagem de banco de dados. Nesse perfil uma tabela é representada como uma classe com o estereotipo <<Tabela>> e um ícone no canto direito superior, conforme a Figura 15. As colunas do banco de dados podem ser modeladas como atributos da classe e descritas as operações estereotipadas associadas com as colunas. Um relacionamento é representado como uma associação estereotipada e inclui um conjunto de chaves primárias e estrangeiras, conforme a Figura 16 (SPARKS, 2002).

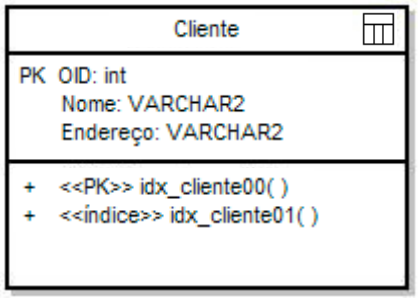

Figura 15 - Classe com o estereótipo tabela
(SPARKS, 2002)

Figura 16 - Associação estereotipada
(SPARKS, 2002)

Usando o tipo de notação descrita na Figura 16, é possível modelar complexas estruturas de dados, mas que atendam, particularmente, o projeto lógico do banco de dados já que não são detalhadas as estruturas físicas de armazenamento. Quanto ao projeto físico, a UML fornece apenas mecanismos que utilizam um componente estereotipado que pode ser visto como uma parte do hardware, conforme a Figura 17.

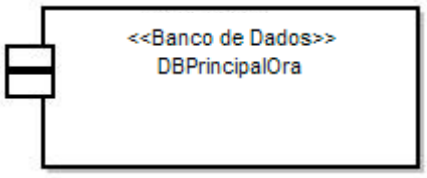

Figura 17 - Entidade Componente Estereotipada
(SPARKS, 2002)

2.6 MANUTENÇÃO DE BANCO DE DADOS

Muitos especialistas em performance de sistemas de informações afirmam que o melhor caminho para o atendimento dos requisitos de performance é o ajuste no código SQL do aplicativo. Porém, quando o aplicativo é bem desenvolvido, as limitações no servidor de banco de dados, incluindo-se o próprio banco de dados, podem ser as causas críticas para que o sistema todo não esteja atingindo os requisitos de performance (ORACLE VIEW, 2004).

Após a implementação de um banco de dados deve-se eleger um profissional ou um grupo de profissionais que possam manipular esse banco. Esses profissionais são conhecidos como DBA´s (ORACLE DOCUMENTATION LIBRARY, 2004).

Dentre as responsabilidades inerentes às tarefas do DBA podem ser citadas, notadamente, as seguintes:

- Instalação e atualização das aplicações e banco de dados corporativos, como ERP e Oracle, respectivamente.
- Alocação do sistema de armazenamento e planejamento futuro dos requisitos de armazenamento para o sistema de banco de dados.
- Definição dos esquemas.
- Criação das estruturas para armazenamento das tabelas (tablespaces).
- Criação dos objetos como tabelas, visões e índices.

- Administração do crescimento dos espaços ocupados em discos pelas tabelas.
- Modificação na estrutura de armazenamento e métodos de acessos do banco de dados.
- Garantia de acesso as informações no banco de dados pelos usuários autorizados.
- Manter o sistema de banco de dados seguro e livre de acessos indevidos.
- Controle e monitoramento dos acessos dos usuários ao banco de dados.
- Monitoramento e otimização da performance do banco de dados.
- Planejamento para a geração das cópias de segurança dos dados e a restauração desses dados no sistema.
- Arquivamento e manutenção das cópias de segurança dos dados.

Alguns fatores importantes na administração do banco de dados e que influenciam no bom desempenho estão relacionados com o crescimento dos arquivos nos discos, a falta de manutenção das tabelas, dos índices e a falta de reorganização dos arquivos. Deve-se haver um constante monitoramento do banco de dados e uma reavaliação contínua das condições de armazenamento dos dados (FINKELSTEIN, SCHKOLNICK & TIBERIO, 1998).

Alguns pontos que devem ser observados na condução de um projeto físico são (ELMASRI & NAVATHE, 2000):

- Tempo de resposta: é o tempo medido entre o envio da transação para o banco de dados pelo usuário até o retorno da resposta. Existem alguns fatores que podem influenciar no tempo de resposta, como: a infra-estrutura da rede de dados, o tempo de acesso do SGBD ao banco de dados, o sistema operacional do servidor do banco de dados, o meio físico utilizado para interligar o servidor do banco de dados ao dispositivo de armazenamento maciço de dados, ou ainda a carga de processamento do sistema.

- Utilização do espaço em disco: é o gerenciamento do espaço em disco utilizado pelos arquivos do banco de dados, incluindo os dados e os índices.

- Performance da transação: é o número médio de transações que podem ser processadas por minuto, geralmente medidas durante os períodos críticos da utilização do sistema.

Esses critérios podem ser avaliados mediante técnicas analíticas ou experimentais, que incluem protótipos ou simulações para se chegar às melhores condições de desempenho do sistema (ELMASRI & NAVATHE, 2000).

2.7 CONSIDERAÇÕES FINAIS

A idéia básica da engenharia de banco de dados é que o sistema seja construído corretamente por meio de uma série de etapas que assegurarão que as necessidades do negócio sejam plenamente atendidas. Ao longo dos anos, as ferramentas de modelagem dos dados e projeto de banco de dados evoluíram consideravelmente ajustando-se a essas necessidades cada vez mais complexas.

Embora o IDEF tenha aproximadamente 30 anos de desenvolvimento apresenta recursos gráficos insuficientes e pouco detalhados para os propósitos de um bom projeto físico de banco de dados.

Por outro lado, a UML é uma linguagem de modelagem recente, mas é fácil de ser aprendida, e com o uso da sua extensão possibilita, ainda, um diagrama gráfico mais elaborado para o projeto físico, além de permitir que os modelos sejam implementados da forma que os projetistas julgarem mais adequados.

Capítulo 3. PROJETO FÍSICO PARA BANCO DE DADOS

3.1 CONSIDERAÇÕES INICIAIS

Um banco de dados relacional não determina automaticamente as melhores organizações de arquivos, armazenamento e métodos de acessos aos dados nos discos. Essas tarefas não são triviais e exigem um bom conhecimento didático e empírico para que sejam tomadas as melhores decisões, já que as conseqüências de manutenção e performance podem ser custosas demais para o negócio.

A seguir, são apresentados os elementos que compõem o projeto físico de um banco de dados, descrevendo, em cada categoria, organização e indexação, bem como algumas das opções existentes.

3.2 ELEMENTOS DO PROJETO FÍSICO

As implementações em banco de dados relacional organizam as linhas das relações em registros nos arquivos e fornece mecanismos de pesquisa sobre as várias colunas das relações. Logo, para desempenho eficiente, o projeto físico de banco de dados deve abranger o estudo das melhores estruturas de armazenamento e recuperação dos dados. Isso inclui os vários elementos do projeto físico como as opções de organização física e técnicas apropriadas de otimização de pesquisa, além dos blocos de dados (WARD & GRIFFITHS, 1999).

Embora a IDEF1X tenha sido amplamente utilizada para a modelagem dos dados, em banco de dados relacional, é mais recomendada para o projeto lógico, não sendo a melhor opção para a implementação não-relacional.

Por outro lado, a UML pode representar toda a estrutura física do banco de dados e usa um componente estereotipado para representar um banco de dados físico. Ainda que não detalhe as organizações e indexações dos

arquivos e os tamanhos dos blocos de dados, ela pode ser estendida com o uso dos perfis para atender esse nível da engenharia de banco de dados.

3.3 ORGANIZAÇÃO DE ARQUIVOS

A organização de arquivos implica em como os registros serão organizados nos blocos físicos em um arquivo do disco. Os blocos são de tamanho único e fixo, determinado pelo sistema operacional. Os registros, entretanto, podem variar de tamanho. A escolha de uma organização de arquivos deve implicar no aumento da eficiência na recuperação dos dados. Os principais tipos de organização de arquivos são: desordenada (*heap*), ordenada (*sequential*), e função *hash* (MOLINA, ULLMAN & WIDOM, 2000).

Esses tipos de organização serão detalhadas e exemplificadas abaixo usando os seguintes dados de estudantes: id, nome, departamento, com um tamanho de bloco de 3 registros por bloco e um arquivo com tamanho total de n blocos (RAMAKRISHNAN, 2003).

11 Chang PH,	07 Ling CS,
04 Brown CE,	03 Gioia PH,
05 White PH,	01 Young CS,
10 Smith CS,	08 Johns PH,
06 Black CE,	09 Jones CE,
02 Hurley CS,	

3.3.1 DESORDENADA

É o tipo mais simples e básico de organização de arquivos. Os registros são armazenados aleatoriamente em qualquer lugar do arquivo, mas, geralmente, são inseridos no final. Como não há uma ordem para os registros, ocorre que a remoção de linhas gera fragmentação no arquivo diminuindo o desempenho. Esse tipo de organização requer pesquisa linear sendo, portanto, utilizada quando se faz uma pesquisa completa em todos os registros do arquivo. A

Figura 18 exibe um exemplo desse tipo de organização referente à relação de estudantes citados (RAMAKRISHNAN, 2003):

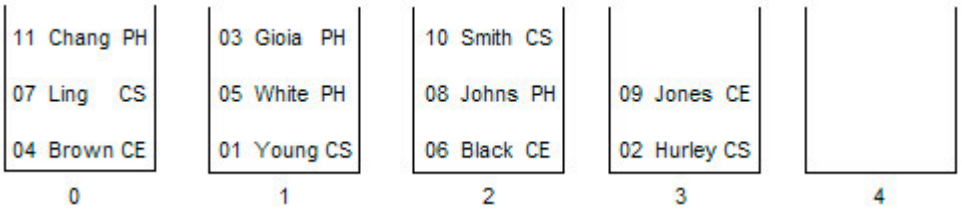

Figura 18 - Organização Desordenada
(RAMAKRISHNAN, 2003)

3.3.2 ORDENADA

Esse tipo de organização de arquivos possibilita que os registros sejam gravados numa ordem ou seqüência determinada pela chave de pesquisa. Geralmente, a chave de pesquisa é a chave primária, embora, isso não seja, necessariamente, uma regra. É recomendada a reorganização periódica dos arquivos para que seja mantida a seqüência dos registros nos discos. Esse tipo de organização suporta pesquisa binária e é mais utilizado quando se pretende recuperar os registros numa ordem específica ou somente um intervalo de registros. A Figura 19 apresenta um exemplo desse tipo de organização referente a relação de estudantes citados (RAMAKRISHNAN, 2003).

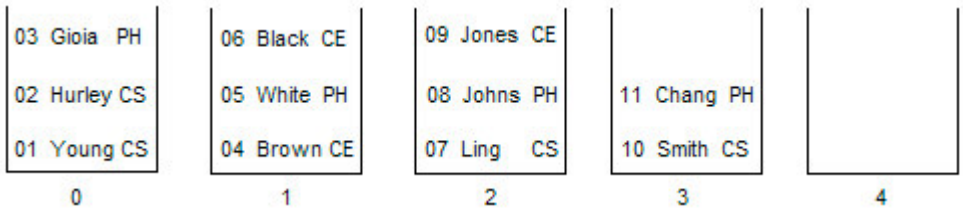

Figura 19 - Organização Ordenada
(RAMAKRISHNAN, 2003)

3.3.3 HASH

A organização *hash* utiliza uma função de cálculo de endereço sob a chave de pesquisa, para determinar o bloco no arquivo onde o registro será armazenado.

Assim, o acesso ao bloco será feito sem a necessidade de uma estrutura de índice. É recomendado para pesquisas exatas, não sendo indicado para pesquisas com intervalos. A seguir, um exemplo desse tipo de organização referente à relação de estudantes citados (RAMAKRISHNAN, 2003):

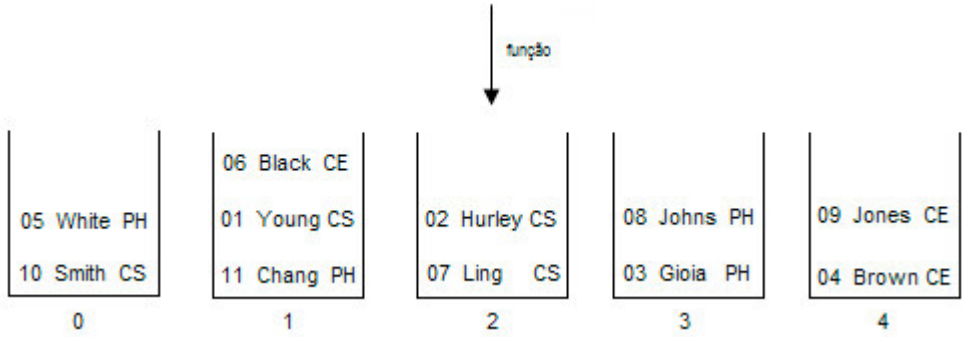

Figura 20 - Organização Hash
(RAMAKRISHNAN, 2003)

A Tabela 3 apresenta as vantagens e desvantagens no uso dos diferentes tipos de organização de arquivos.

Tabela 3 - Organização de Arquivos
(RAMAKRISHNAN, 2003)

Organização	Vantagens	Desvantagens
Desordenada	• Inserção rápida de registros no final do arquivo.	• Deleção de registros gera fragmentação. • Pesquisa de dados lenta, pois requer uma busca linear em todo o arquivo.
Ordenada	• Pesquisa dos dados rápida com o uso da pesquisa binária.	• A inserção de novos registros movimenta uma grande quantidade de registros para que seja mantida a ordem de organização. • Inserção e remoção lentas. • Pesquisa de dados lenta quando feita pelo campo que não é a chave de ordenação. • Organização periódica dos arquivos para preenchimento dos espaços

		dos registros removidos.
Hash	• Pesquisa muito rápida quando se utiliza a função hash.	• Não recomendado para registros ordenados ou pesquisas com intervalos.

3.4 ESTRUTURAS DE INDEXAÇÃO

As estruturas de indexação são métodos eficientes de acesso aos dados no disco baseados na organização dos arquivos. Um banco de dados relacional puro não é capaz de recuperar os dados das linhas em uma tabela de maneira eficiente sem o auxílio das estruturas de indexação (SRIVASTAVA & NGO, 1999). Um arquivo indexado por campo e ponteiros melhora o desempenho de localização dos dados baseado no valor de um campo, conhecido como campo de indexação, ou atributo de indexação, sem a necessidade de busca completa em toda a tabela. O número de acesso ao disco também é reduzido (MEMON, 2004) (MOORE, 1996).

Os índices são criados explícita ou automaticamente e são estruturas independentes da tabela que indexam, pois podem ser criados ou removidos a qualquer momento sem efeito nas tabelas ou em outros índices. Porém, deve-se considerar que toda a modificação ocorrida num arquivo implicará também na modificação de cada índice para aquele arquivo (SRIVASTAVA & NGO, 1999) (ULLMAN, 1998).

Os tipos de índices mais usados são baseados em: índice primário (arquivos seqüenciais), índice secundário, índice agrupamento (*clustering*), índices árvore B, índice mapa de bits (*bitmap*) e índice reverso (*reverse*).

3.4.1 ÍNDICE PRIMÁRIO

A estrutura de índice sobre arquivos seqüenciais é uma das mais simples. Um arquivo de dados classificado recebe um arquivo de índice baseado em pares chave-ponteiro. Essa estrutura é também conhecida como índice primário porque é particularmente utilizada quando a chave de pesquisa é a chave primária da relação (MOLINA, ULLMAN & WIDOM, 2000).

O índice primário é um arquivo ordenado que armazena a chave primária e um ponteiro para o bloco de dados no disco. Essa chave é também a chave de pesquisa e determina a ordem seqüencial do arquivo no disco. Cada entrada no índice tem o valor da chave primária para o primeiro registro daquele bloco de dados do arquivo e também um ponteiro para o bloco. É utilizado somente quando o campo de indexação é uma chave primária e os dados do arquivo estão organizados por essa chave (MEMON, 2004). Pode ser esparso ou denso. A Figura 21a exemplifica um índice denso (à esquerda) sobre um arquivo de dados seqüenciais (à direita), onde há uma entrada no índice para cada um dos registros do arquivo de dados. Um índice esparso, conforme a Figura 21b, tem uma entrada no índice para somente alguns registros (MOLINA, ULLMAN & WIDOM, 2000).

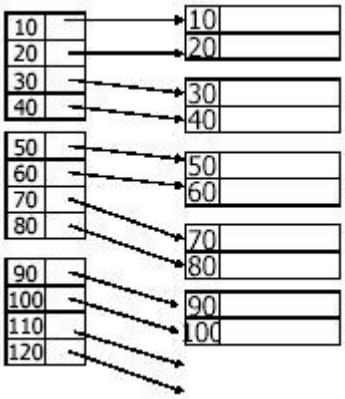

Figura 21a - Índice Primário Denso (MOLINA, ULLMAN & WIDOM, 2000)

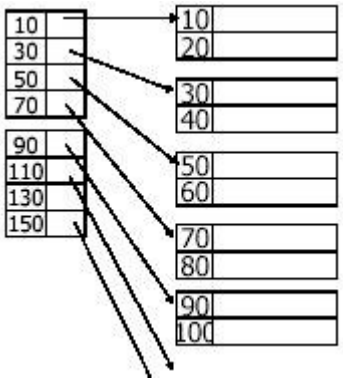

Figura 21b - Índice Primário Esparso (MOLINA, ULLMAN & WIDOM, 2000)

3.4.2 ÍNDICE SECUNDÁRIO

O índice secundário é um arquivo ordenado que armazena a chave secundária e um ponteiro, porém, essa não é a ordenação do arquivo de dados indexado. É utilizado sobre um arquivo de dados que não está organizado pelo campo de indexação, ou chave primária, como é o caso de arquivos desordenados. Por isso, o índice secundário precisa ser sempre denso. A Figura 22 exemplifica um índice secundário denso sobre um campo chave não ordenado de um arquivo.

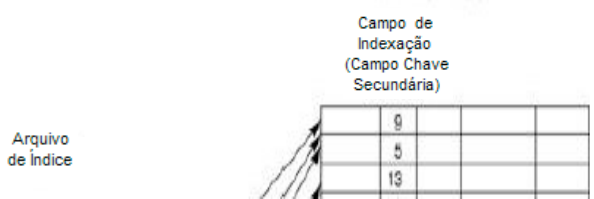

Figura 22 - Índice Secundário
(MEMON, 2004)

3.4.3 ÍNDICE AGRUPAMENTO

O índice agrupamento (*clustering*) é parecido com o índice primário, porém é um arquivo ordenado por um campo que não é chave e também possui um ponteiro para o bloco de dados. Os registros no arquivo de dados estão ordenados pelo campo chave. Podem ser mais lentos que os índices primários em pesquisas que recuperam um grande número de registros seqüenciais, conforme a Figura 23.

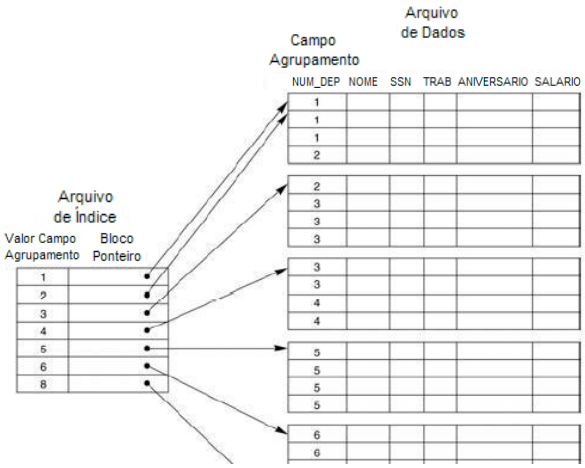

Figura 23 - Índice Agrupamento
(MEMON, 2004)

3.4.4 ÍNDICES ÁRVORE B

Os bancos de dados crescem pouco ou muito ao longo do tempo e, conforme os arquivos aumentam, a performance dos índices ordenados piora

gradativamente. Esse problema pode ser ligeiramente resolvido com a organização periódica dos arquivos. Porém, durante a organização nenhum usuário poderá acessar o sistema, ou seja, os dados estarão indisponíveis até que a organização esteja completa. Assim, uma melhor opção pode ser uma das duas estruturas de índices árvore B que se baseiam na estrutura de árvore balanceada em que todos os dados estão no mesmo nível. Isto quer dizer que, o caminho da raiz da árvore até as suas folhas, onde estão os dados, é do mesmo comprimento, como se fosse uma árvore balanceada e todas as folhas possuem ponteiros para o arquivo de dados na tabela indexada. Como é uma árvore balanceada é um método de acesso interessante para pesquisas aleatórias sobre as chaves primárias ou arquivos seqüenciais. Em geral, os índices árvore B controlam automaticamente os vários níveis de índices que sejam necessários para o tamanho do arquivo indexado. Além disso, gerenciam o espaço nos blocos do disco para que não fiquem cheios e não ocorra nenhum estouro de bloco para o índice (MOLINA, ULLMAN & WIDOM, 2000).

Um índice B-Tree é um conjunto ordenado de entradas e que contém um valor chave de pesquisa e um ponteiro para uma linha específica na coluna indexada. Uma vez que o valor chave é localizado o ponteiro identifica a linha correspondente na tabela. Comumente essa estrutura de índice é menor que a tabela e é mais recomendado sobre as colunas que possuem valores únicos, como uma coluna ID_Cliente que, no geral, é um valor único para identificar cada cliente. O B-Tree também é mais eficiente quando a pesquisa recupera menos que 20% das linhas de uma tabela. Acima disso, a melhor opção é uma pesquisa seqüencial na tabela. Quando uma pesquisa solicita somente valores cobertos por um índice, a resposta pode ocorrer sem o acesso à tabela. Por exemplo, uma pesquisa que solicite a contagem de quantas linhas da tabela são referentes ao funcionário de número 114956 pode obter a resposta através da coluna ID do índice contando as entradas que possuem esse número como valor chave de pesquisa, sem mesmo ler uma única linha da tabela. Por outro lado, se a mesma pesquisa solicitar o nome do funcionário e não houver nenhum índice para a coluna Nome, então o índice localizará na tabela as linhas que contém o número 114956 para recuperar o nome. Mesmo assim, a

pesquisa por índice é melhor que a pesquisa na tabela quando há poucas linhas envolvidas (IBM, 2004).

Deve-se considerar, ainda, que o índice B-Tree elimina a redundância das chaves de pesquisa ocorridas no B^+-Tree e o seu número de nós também é reduzido, melhorando em, no mínimo, 50% o aproveitamento no dispositivo de armazenamento. Por outro lado, sua implementação é mais complicada e somente uma parte das chaves de pesquisa são encontradas mais rapidamente (MOORE, 1996). A Figura 24 ilustra um B-Tree com três níveis a partir da raiz.

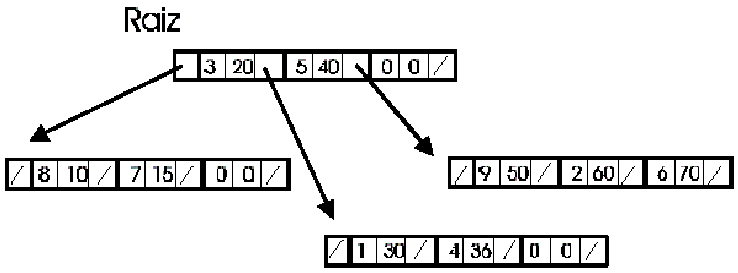

Figura 24 - Índice Árvore B
(MOORE, 1996)

O B^+-Tree é uma estrutura de indexação variante do B-Tree. Nessa estrutura ocorre a redundância das chaves de pesquisa nos diferentes níveis da árvore para a otimização no acesso aos registros no arquivo de dados. A Figura 25 ilustra uma exemplificação genérica.

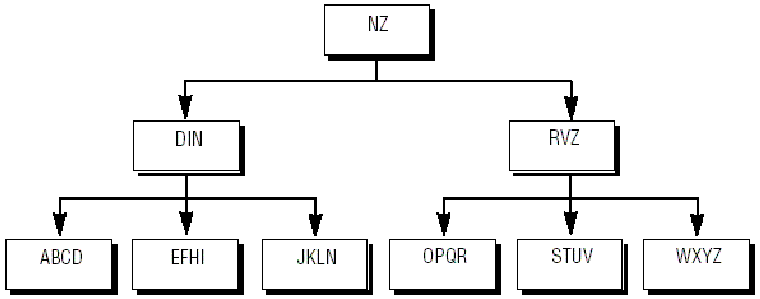

Figura 25 - Índice Árvore B^+ Genérico
(MOORE, 1996)

As chaves N e Z estão armazenadas em todos os níveis da árvore. As chaves D, I, R e V também aparecem na folha subseqüente. A repetição das chaves

mais altas, nas folhas mais baixas indica uma organização de chave ascendente.

Para realizar uma busca que localize a letra H, a pesquisa começará pela raiz NZ usando a pesquisa binária da seguinte forma:

- A letra H é menor ou igual a N? Menor, portanto desça para o nível seguinte no lado esquerdo da árvore.
- A letra H é menor ou igual a D? Maior.
- A letra H é menor ou igual a I? Menor, portanto desça para as folhas EFHI e encontre H.

A Figura 26 ilustra um exemplo de índice árvore B^+.

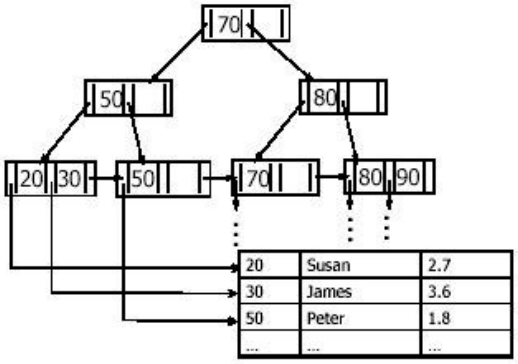

Figura 26 - Índice Árvore B^+
(MEMON, 2004)

Uma diferença significativa entre o índice árvore e o índice primário é que o índice árvore aponta para uma folha de dados enquanto o índice primário possui ponteiros para os blocos de dados na tabela.

3.4.5 ÍNDICE MAPA DE BITS

O índice mapa de bits (*bitmap*), é uma estrutura de indexação que utiliza um bit, 0 ou 1, para cada um dos diferentes valores da coluna da tabela indexada. O valor armazenado no índice será 1 para indicar que há correspondente na coluna da tabela (ou valor verdadeiro) e será 0 caso não haja correspondente (ou valor falso) (ORACLE, 2004). Logo, esse índice deve ser utilizado somente

em situações onde existam poucos valores distintos na coluna indexada. A Tabela 4 ilustra um exemplo dessa estrutura de índice em que todas as colunas são de baixa cardinalidade porque possuem poucos valores distintos (ORACLE, 2002).

Tabela 4 - Cliente
(ORACLE, 2002)

ROWID	Estado_Civil	Região	Sexo	Nível de Suporte
101	Solteiro	Leste	Masculino	Suporte_1
102	Casada	Central	Feminino	Suporte_4
103	Casada	Oeste	Feminino	Suporte_2
104	Divorciado	Oeste	Masculino	Suporte_4

O índice *bitmap* para a coluna *Região* será constituído de três mapas de bits, conforme a Tabela 5, sendo um para cada região, mais o *ROWID* de cada linha da tabela.

Tabela 5 - Mapa de Bits da Tabela Cliente
(ORACLE, 2002)

Cliente	Região: 'Leste'	Região: 'Central'	Região: 'Oeste'
101	1	0	0
102	0	1	0
103	0	0	1
104	0	0	1

Uma das vantagens do mapa de bits sobre a árvore binária é a redução de espaço em disco caso a coluna da tabela indexada possua muitos valores repetidos. E, também, como o índice mapa de bits é armazenado no formato comprimido a redução de espaço é de 25%, no mínimo (LONEY & THERIAULT, 2000) (ORACLE, 2002).

No caso de múltiplas colunas que precisam ser indexadas, o índice mapa de bits, também pode ser mais vantajoso. Em bancos de dados que contenham

somente índices árvore binária deve-se antecipar as colunas que serão acessadas juntas numa única pesquisa e criar um índice composto para elas. No exemplo da tabela Cliente, um índice B^+-Tree nas colunas *Estado_Civil*, *Região* e *Sexo*, é menos útil para as pesquisas que acessam somente *Região* e *Sexo*. Para completamente indexar o banco de dados, deve-se criar índices sobre outras permutações dessas colunas. Num simples caso de três colunas com baixa cardinalidade, há seis possíveis índices B^+-Tree compostos que consumirão espaço em disco e performance. O mapa de bits pode ser mais eficiente porque somente três pequenos índices *bitmap* de única coluna farão o trabalho de seis índices árvore B^+ de três colunas (ORACLE, 2002).

3.4.6 ÍNDICE REVERSO

O índice reverso é um arquivo que armazena os bytes de cada coluna indexada na forma reversa, com exceção do *ROWID*. Por exemplo, os valores 1234 tornam-se 4321 no índice. É particularmente recomendado nas situações em que os usuários inserem valores ascendentes e os mais antigos (ou baixos) são removidos. No caso do banco de dados Oracle esse tipo de índice é mais restrito ao ambiente com servidor de processamento paralelo. No caso de um banco de dados Oracle 8, um índice reverso é identificado pelo valor 0x04 na coluna Propriedade da tabela *view* $ind, conforme segue abaixo (ORACLE, 2002):

OBJ#	*DATAOBJ#*	*TYPE#*	*PROPRIEDADE*	
24051	*24051*	*1*	*1*	
24053	*24053*	*1*	*1*	
24071	*24071*	*1*	***4***	← índice reverso

3.4.7 ÍNDICE FUNÇÃO HASH

O índice função *hash* organiza as chaves de pesquisa com os seus ponteiros de registro associados a uma estrutura de arquivos *hash*. Pode ser utilizado tanto para a organização de arquivo quanto para a estruturação de índice. Os índices *hash* são sempre índices secundários porque se uma tabela de dados está organizada por *hash* torna-se desnecessário um índice primário usando a mesma chave de pesquisa, ou seja, os índices são índices secundários para

arquivos organizados por *hash* (SILBERSCHATZ, KORTH & SUDARSHAN, 2002). A Figura 27 apresenta um exemplo.

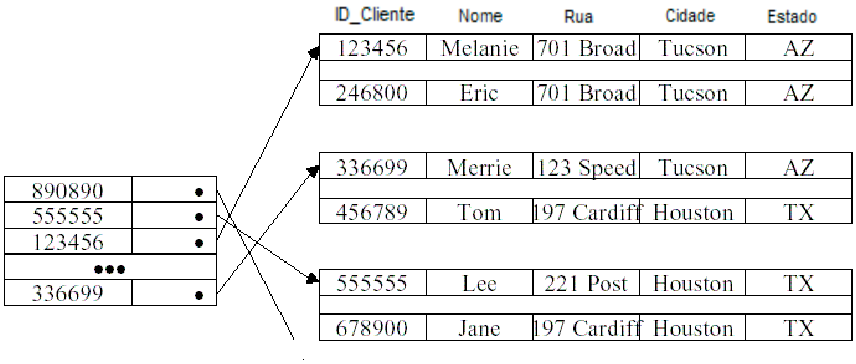

Figura 27 - Índice Função *Hash*
(ORACLE, 2002)

Capítulo 4. EXTENSÃO DO UML PROFILE DE BANCO DE DADOS PARA PROJETO FÍSICO

4.1 CONSIDERAÇÕES INICIAIS

Diante da quantidade e complexidade dos elementos que compõem o projeto físico de um banco de dados, na concepção de uma extensão para o modelo de representação física, é necessária, inicialmente, uma definição de ícones para esses elementos.

Notadamente, o projeto físico para banco de dados, visa atender a demanda de um ótimo tempo de resposta mesmo diante de um alto volume de transações.

4.2 APRESENTAÇÃO DA PROPOSTA

A definição para a extensão do *UML profile* de banco de dados para projeto físico é, inicialmente, apresentada na Tabela 6, resultado do refinamento das pesquisas realizadas. A extensão está dividida em tipos de índices e organização dos arquivos. Para cada um dos tipos são apresentados os estereótipos ou os ícones correspondentes que poderão ser utilizados para o projeto físico (AVANZI & CAMOLESI Jr, 2004).

Uma das vantagens da sua utilização é a redução na complexidade de projetos em grandes bancos de dados onde, notadamente, ocorrem os maiores problemas de desempenho dos sistemas (MOK & PETER, 2001). Essa tabela pode ser estendida para atender outros índices (ELMASRI & NAVATHE, 2000), (MOLINA, ULLMAN & WIDOM, 2000).

Tabela 6 - Estereótipos e Ícones
(AVANZI & CAMOLESI Jr, 2004)

		Estereótipo	Ícone
Índice	Primário Denso	<<PD>>	
	Primário Esparso	<<PS>>	
	Agrupamento	<<CS>>	
	Secundário	<<SC>>	
	Árvore B	<<BT>>	
	Árvore B^+	<<BT^+>>	
	Mapa de Bits	<<MB>>	
	Reverso	<<RV>>	
Organização	Desordenado	<<heap>>	
	Ordenado	<<sequential>>	
	Função	<<hash>>	

Para a utilização dos estereótipos e ícones da Tabela 6, deve-se adotar uma das seguintes alternativas abaixo (AVANZI & CAMOLESI Jr, 2004):

- Utilização exclusiva dos estereótipos ou ícones da Tabela 6 para a representação dos índices e organização dos arquivos.
- Utilização dos estereótipos da Tabela 6 e uso do ícone padrão na UML.
- Utilização do ícone padrão na UML e assumir que a organização do arquivo é aquela suportada pelo gerenciador do banco de dados.

A Figura 28 ilustra um exemplo genérico de uma classe na qual podemos verificar em detalhes a aplicação da Tabela 6 com a especificação da chave de organização, a organização do arquivo e a indexação. No nome da classe há a descrição do estereótipo e um ícone que podem ser utilizados para representar a organização do arquivo. Nesse caso, o ícone indica que a organização do arquivo é *sequential* ordenado pela chave TITULO. Algumas organizações não

necessitam de chave, como a *heap*, ao contrário de outras em que ela é obrigatória. Os atributos AUTOR e ISBN são especificados como índices B^+-Tree usando o estereótipo <<BT^+>> (poderia ser utilizado o ícone da Tabela 6) os quais estão sendo indexados pelos índices ordenados IX_AUTOR( ) e IX_ISBN( ).

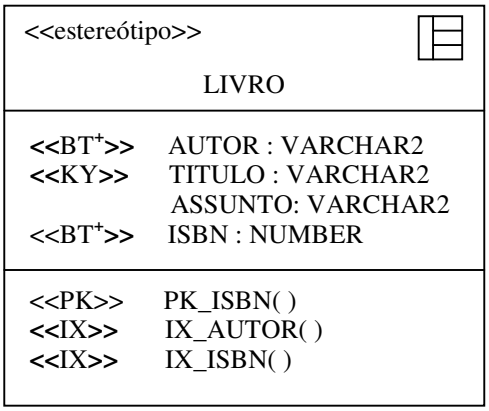

Figura 28 - Exemplo Genérico de Classe
(AVANZI & CAMOLESI Jr, 2004)

Na Figura 29 é ilustrado um exemplo para o gerenciamento dos pedidos de compras em algum sistema de informação. As tabelas Pedido, Item_Pedido, Vendedor e Cliente podem ser especificadas conforme a Tabela 6 para implementação no nível físico. Nesse exemplo foram utilizados os estereótipos e ícones da Tabela 6. A tabela PEDIDO possui um ícone no nome da classe identificando que a organização do arquivo é ordenada. O estereotipo <<KY>> indica que o atributo DATA_PEDIDO foi utilizado como chave para essa organização. Os ícones presentes nos atributos ID_CLIENTE e ID_SOLICITA indicam que eles são índices B+-Tree. Esses índices estão indexados por IX_ID_CLIENTE( ) e IX_ID_SOLICITA( ), respectivamente, conforme os estereótipos <<IX>>[2]. As chaves primárias e estrangeiras estão identificadas pelos estereótipos <<PK>> e <<FK>> conforme *UML profile* de banco de dados já existente. A tabela ITEM_PEDIDO possui uma organização de arquivo ordenada indicada pelo estereótipo <<sequential>> e está organizada pela chave COD_UNIT. A indexação *clustering* está representada por um ícone no atributo COD_COMPONENTE o qual está indexado por IX_COD_

COMPONENTE( ), conforme representado pelo estereótipo <<IX>>. A indexação *clustering* foi escolhida já que os componentes podem ser agrupados por itens. A organização de arquivo na tabela VENDEDOR é desorganizada, representada por um ícone, portanto, não apresenta chave de organização. A tabela CLIENTE possui uma organização de arquivo por função identificada pelo estereótipo <<hash>>. A chave de organização dessa tabela é ID_LISTA_PRECO representada pelo estereótipo <<KY>>. O índice NOME_CLIENTE é B+-Tree representado por um ícone e está indexado por IX_NOME_CLIENTE( ) conforme o estereótipo <<IX>>.

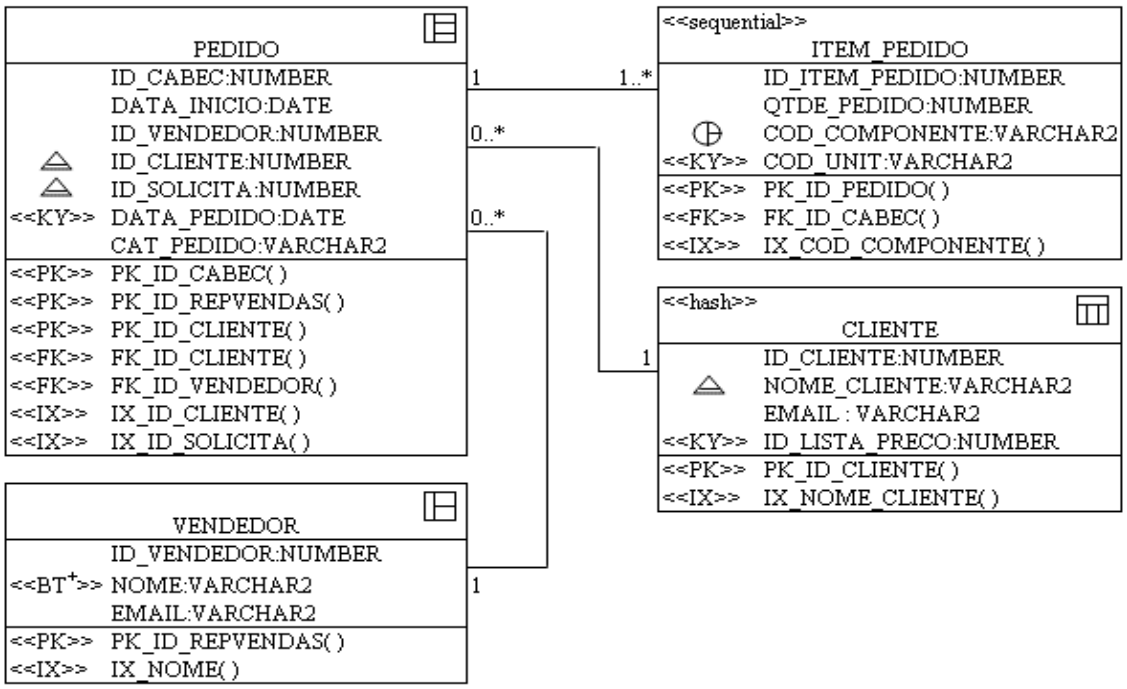

Figura 29 - Exemplo Genérico de Classe
(AVANZI & CAMOLESI Jr, 2004)

A Figura 30 ilustra parte dos diagramas de nós e componentes no qual é especificado o tamanho dos blocos de dados. O tamanho do bloco de dados é definido em cada servidor para melhor desempenho durante o processamento. Esses diagramas também apresentam os tamanhos dos *caches*, importantes para o desempenho bom de um sistema.

[2] PK, FK e IX já estão presentes no *UML profile.*

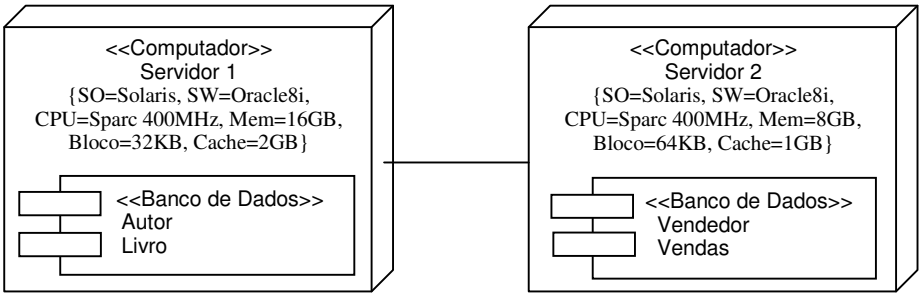

Figura 30 - Diagramas de Nós e Componentes
(AVANZI & CAMOLESI Jr, 2004)

4.3 REQUISITOS PARA O PROJETO FÍSICO DO BANCO DE DADOS

Para minimamente assegurar que as escolhas realizadas no projeto físico surtam os efeitos de desempenho almejados, são necessárias modelagens específicas de aspectos relacionados ao uso dos dados pelos usuários. Esta seção apresenta uma proposta desta modelagem, com base em modelos tradicionais da UML.

4.3.1 DIAGRAMA DE CASO DE USO

O diagrama de caso de uso (OMG, 2003) preocupa-se em representar as funcionalidades de um sistema. A Figura 31a ilustra o exemplo de diagrama de caso de uso para o usuário Operador-1 que acessa o Sistema COSC (Controle das Operações nos Servidores Corporativos) para solicitar a pesquisa do nome do BUC (Boletim de Utilização do Computador).

Sabemos que sistemas diferentes podem concorrer com as mesmas tabelas no banco de dados. Assim, a Figura 31b ilustra o exemplo de diagrama de caso de uso para o usuário Analista que utiliza o sistema AUSC (Análise no Uso dos Servidores Corporativos) para conseguir acesso à mesma tabela do sistema COSC.

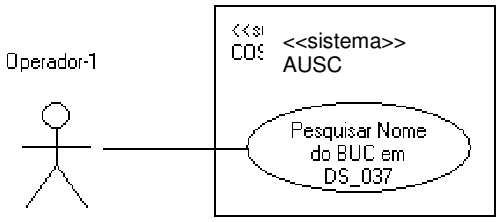

Figura 31a - Exemplo de Caso de Uso para o Operador-1

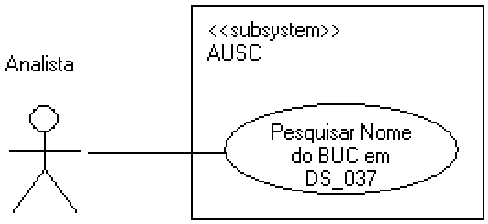

Figura 31b - Exemplo de Caso de Uso para o Analista

Para obtermos um projeto com desempenho otimizado precisamos considerar a freqüência com que isso estará ocorrendo. Desta forma, para complementar o diagrama de casos de uso se faz necessária uma Tabela de Freqüência, conforme a Tabela 7, na qual são especificados os detalhes da freqüência de execução. A coluna Sistema identifica todos os sistemas que acessam o banco de dados. A coluna Caso de Uso indica todos os casos de uso modelados nos sistemas. Na coluna Freqüência é especificada a freqüência de execução de cada caso de uso, utilizando granularidade de tempo: minuto, hora, dia, mês ou ano.

Tabela 7 - Freqüência para Caso de Uso
(AVANZI & CAMOLESI Jr, 2004)

Sistema	Caso de Uso	Freqüência
COSC	Pesquisa de BUC	200/dia
AUSC	Pesquisa de BUC	2/mês
COSC	Análise de BUC	20/dia

4.3.2 DIAGRAMA DE SEQÜÊNCIA E O PROJETO LÓGICO DO BANCO DE DADOS

Antecedendo o projeto físico de banco de dados, para poder tomar decisões sobre as organizações e indexações mais adequadas para as tabelas, é necessário especificar os acessos as tabelas. A representação explícita e detalhada de comandos SQL em diagramas de Seqüência (OMG, 2003) é uma forma eficiente para o reconhecimento dessas operações.

A Figura 32 apresenta um diagrama de Seqüência UML onde apresentamos também os acessos ao banco de dados. Nesse exemplo é ilustrado como representar cada acesso feito às tabelas do banco de dados. A diferença em relação ao UML padrão é que os comandos de manipulação SQL estão declarados em cada tabela. Geralmente, nada disso é incluído em um diagrama de Seqüência. Deve haver um diagrama de Seqüência para cada Caso de Uso.

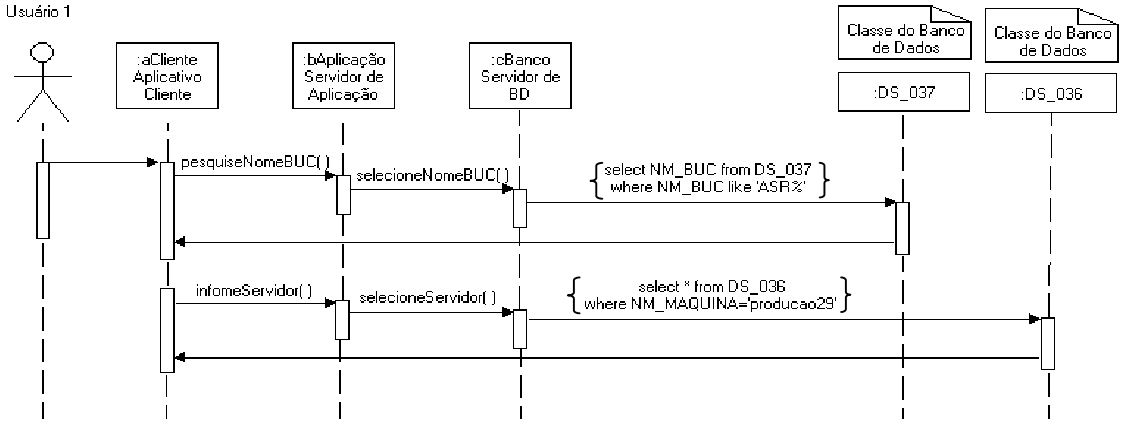

Figura 32 - Diagrama de Seqüência com Comandos SQL
(AVANZI & CAMOLESI Jr, 2004)

4.3.3 TOMANDO DECISÕES DE PROJETO

Os diagramas de seqüência existentes para os casos de uso que acessam o banco de dados podem apresentar necessidade por organizações divergentes causando a tomada de decisões de projeto. Por exemplo, o caso de uso Pesquisa de BUC, do Sistema COSC, que realiza uma consulta (*select*) sobre a tabela DS_037 e se beneficiaria de uma organização por função pela coluna ID_BUC. No entanto, o caso de uso Pesquisa de BUC, do Sistema AUSC, que também realiza uma consulta (*select*) sobre a tabela DS_037, mas se beneficiaria por uma organização seqüencial pela coluna ID_RESPONSAVEL. Temos, desta forma, um conflito de interesses, cuja solução passa pela análise de prioridade de execução otimizada entre os dois casos de uso. A lógica de priorização de caso de uso pode considerar a freqüência de execução, conforme a Tabela 7, priorizando assim, casos de uso mais freqüentes na escolha pela organização.

O engenheiro ou administrador do banco de dados pode estabelecer outra lógica de priorização se verificadas que outras questões devem ser consideradas como, por exemplo, políticas e imposições externas. Independente da lógica empregada, a equipe de engenharia e administração de um banco de dados terá a especificação de freqüências de casos de uso e os respectivos diagramas de seqüência com comandos SQL, que permitirão a identificação das repercussões de suas decisões sobre os aspectos físicos.

4.4 CONSIDERAÇÕES FINAIS

Este capítulo descreveu e apresentou exemplos da proposta da extensão do *UML Profile* para projeto físico de banco de dados e o uso dos diagramas tradicionais da UML como a tabela de freqüência para caso de uso e a especificação de diagrama de seqüência com comandos SQL para as tomadas de decisões de projeto.

O emprego da Tabela 6 (Estereótipos e Ícones) na modelagem dos projetos permite que os projetistas especifiquem, detalhadamente, os aspectos físicos importantes de desempenho de banco de dados. Diante dessa exposição, é possível sustentar as colocações e o trabalho apresentado no próximo capítulo.

Capítulo 5. ESTUDO DE CASO

5.1 CONSIDERAÇÕES INICIAIS

O presente caso de estudo foi verificado no projeto de um sistema, desenvolvido no Departamento de Sistemas de Informações, da empresa Indústrias Romi S.A., situada em Santa Bárbara d´Oeste. O sistema é conhecido como CONDOR (CONtrole Das Operações Rotinizadas) e administrado pelos operadores do setor Central de Operações, do Departamento de Sistemas de Informações, para apoio aos serviços diários executados por aquele setor no uso dos 32 servidores corporativos, através de BUCs (Boletins de Utilização do Computador).

5.2 EXPLANAÇÃO DO CASO

A empresa utilizava um sistema similar ao CONDOR, desenvolvido em Clipper, para o controle das operações diárias das atividades da Central de Operações, como: salvamento e recuperação do banco de dados Oracle, correio eletrônico, servidores corporativos departamentais, controles das mídias magnéticas, locais de armazenamento, instruções sobre a execução dos aplicativos concorrentes Oracle e outros comandos operacionais nos servidores. Diariamente é realizado o salvamento de 1 TB (*Terabyte*) em mídias magnéticas. Assim, houve a necessidade da atualização do programa e o armazenamento das informações das tarefas e o histórico das ocorrências no banco de dados Oracle.

O projeto CONDOR foi recentemente implementado migrando o antigo sistema para a intranet no portal corporativo *web*, com tecnologia Oracle, dando maior flexibilidade de uso, interface agradável e segurança dos dados que estão agora armazenados em dispositivo de armazenamento de alta disponibilidade. O projeto lógico do sistema CONDOR foi projetado e implementado da forma

que, geralmente, ocorre nesse nível da engenharia de banco de dados, sem levar em consideração os detalhes do projeto físico, conforme a Figura 33.

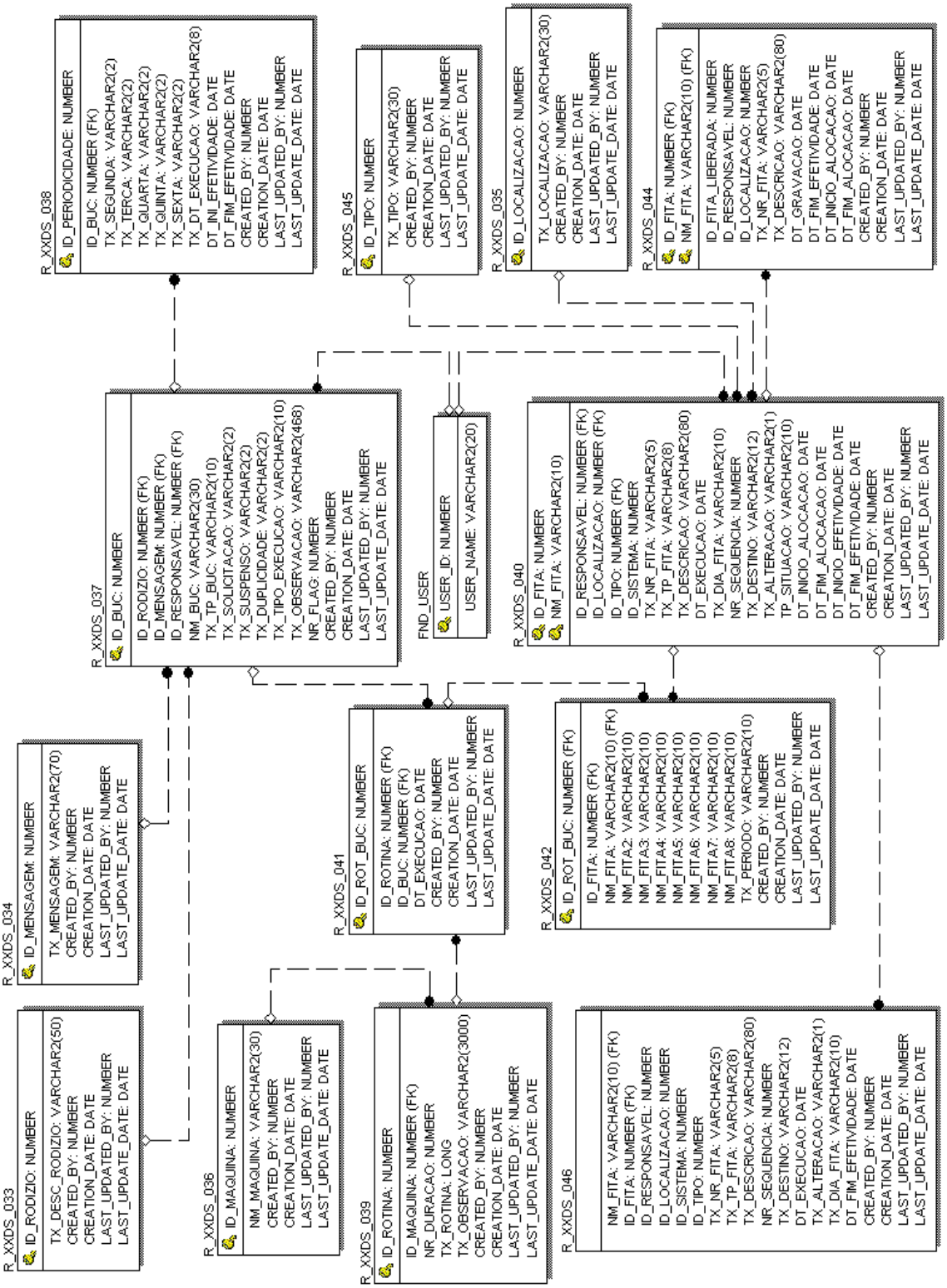

Figura 33 - Esquema Lógico do CONDOR
(DEPARTAMENTO DE SISTEMAS DE INFORMAÇÕES - INDÚSTRIAS ROMI S.A., 2005)

5.3 APLICAÇÃO DO TRABALHO

No projeto apresentado anteriormente não foram consideradas a organização dos arquivos e as estruturas de indexação, exceto os índices B^+-Tree que são criados automaticamente pelo gerenciador do banco de dados Oracle para as chaves primárias. Não raramente, o projeto físico de um banco de dados é pouco valorizado na prática dando possibilidade a futuros problemas de desempenho e administração do banco.

No geral, os procedimentos de implantação das tabelas no banco de dados são sempre os mesmos e seguem as práticas já adotadas pela empresa. Ou seja, comumente, não são pensados em organização de arquivos e as definições de índices são sempre índices árvore B^+.

Assim, o projeto do sistema CONDOR foi então detalhado com as especificações do projeto físico na UML, conforme a Figura 34, visando a melhoria do desempenho no acesso aos dados e a documentação melhorada do projeto.

Para exemplificação desse caso de estudo, foram utilizados, propositadamente, os estereótipos em algumas tabelas e os ícones em outras, não havendo uma padronização. Deve-se, entretanto, adotar um padrão para uso da Tabela 6, ou seja, utilizar somente os estereótipos ou somente os ícones para que não ocorram desordens de entendimento e desestruturação gráfica.

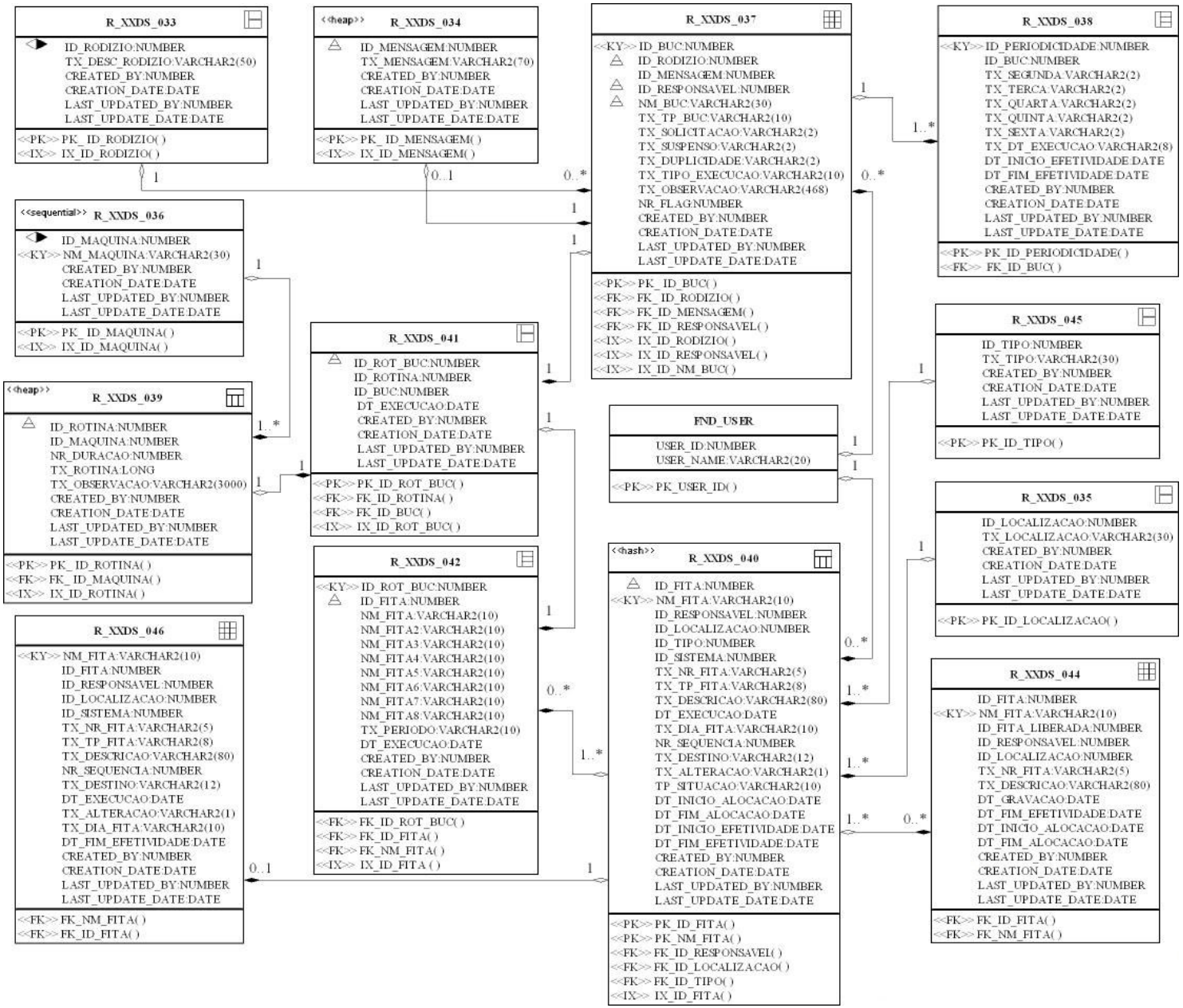

Figura 34 – Esquema Lógico do CONDOR detalhado conforme a Tabela 6
(Elaborado por este autor a partir da Figura 33 da Romi, 2005)

As organizações dos arquivos e índices apresentados no novo projeto foram especificados conforme a Tabela 6 de Estereótipos e Ícones:

R_XXDS_033: É uma tabela para a descrição dos rodízios de salvamento dos dados, como: salvamento diário, semanal, mensal, etc. Não há projeção de crescimento significativo nessa tabela. A tabela possui um ícone no nome da classe indicando que a organização de arquivos é desordenada, não havendo chave de organização. O ícone no atributo ID_RODIZIO indica que é um índice mapa de bits e a tabela está indexada por IX_ID_RODIZIO( ), conforme o estereótipo <<IX>>. Foi utilizado o índice mapa de bits, pois as colunas são de baixa cardinalidade e possuem poucos valores distintos.

R_XXDS_034: É uma tabela de cadastro de informações adicionais que podem ser inseridas no BUC. Não há projeção de crescimento significativo nessa tabela. A tabela possui o estereótipo <<heap>> no nome da classe indicando que a organização de arquivos é desordenada, não havendo chave de organização. O ícone no atributo ID_MENSAGEM indica que é um índice árvore B^+ e a tabela está indexada por IX_ID_MENSAGEM( ), conforme o estereótipo <<IX>>. Foi utilizado o índice árvore B^+ para melhor desempenho no acesso aos dados.

R_XXDS_035: É uma tabela que especifica o local físico de armazenamento das fitas magnéticas, como: cofre, terceiros, etc. Há somente 7 localizações físicas. A tabela possui um ícone no nome da classe indicando que a organização de arquivos é desordenada, não havendo chave de organização. Não foi especificado índice, ficando a critério do gerenciador do banco de dados. Ou seja, nesse caso, a chave primária PK_ID_LOCALIZACAO( ) é o índice primário.

R_XXDS_036: É uma tabela que especifica em qual servidor será executado o BUC. Há 32 servidores corporativos sem previsão de

aquisições significativas em curto prazo. Poderá ocorrer a atualização dos servidores já existentes por outros tecnologicamente mais modernos. A tabela possui o estereótipo <<sequential>> no nome da classe indicando que a organização da tabela é ordenada e está organizada pela chave NM_MAQUINA, conforme o estereótipo <<KY>>. O ícone no atributo ID_MAQUINA indica que é um índice mapa de bits e a tabela está indexada por IX_ID_MAQUINA( ), conforme o estereótipo <<IX>>. Foi utilizado o índice mapa de bits, pois as colunas são de baixa cardinalidade e possuem poucos valores distintos.

R_XXDS_037: É a principal tabela do sistema onde são cadastrados todos os BUCs e crescerá progressivamente com a adição de novas tarefas nas rotinas da Central de Operações. A tabela possui um ícone no nome da classe indicando que a tabela está organizada por função pela chave ID_BUC_NUMBER, conforme o estereótipo <<KY>>. Os ícones nos atributos ID_RODIZIO, ID_RESPONSAVEL e NM_BUC, indicam índices árvores B^+, para melhor desempenho no acesso aos dados, indexados por IX_ ID_RODIZIO( ), IX_ID_ RESPONSAVEL( ) e IX_ID_NM_BUC( ), conforme os estereótipos <<IX>>.

R_XXDS_038: É uma tabela contendo períodos pré-definidos de execução durante a semana informando quando cada BUC será executado. A tabela possui um ícone no nome da classe indicando que a organização de arquivos é ordenada pela chave primária ID_PERIODICIDADE, conforme o estereótipo <<KY>>. Não foi especificado índice, ficando a critério do gerenciador do banco de dados. Ou seja, nesse caso, a chave primária PK_ID_ PERIODICIDADE( ) é o índice primário.

R_XXDS_039: É uma tabela utilizada para a descrição textual das rotinas do BUC não exigindo uma organização complexa. Exemplo:

executar salvamento completo do servidor STARGATE. A tabela possui o estereótipo <<heap>> no nome da classe indicando que a organização é desordenada, portanto, não há chave de organização. O ícone no atributo ID_ROTINA indica que é índice árvore B^+ indexado por IX_ID_ROTINA( ).

R_XXDS_040: É uma das maiores tabelas sendo responsável pelo cadastro de todas as mídias magnéticas. Na geração dos BUCs as datas dessa tabela serão alteradas. A tabela possui o estereótipo <<hash>> no nome da classe indicando que a organização é por função pela chave NM_FITA, conforme o estereótipo <<KY>>. O ícone no atributo ID_FITA indica que é um índice árvore B^+ e a tabela está indexada por IX_ID_FITA( ), conforme o estereótipo <<IX>>. Foi utilizado o índice árvore B^+ para melhor desempenho no acesso aos dados.

R_XXDS_041: É uma tabela de relacionamento entre as tabelas R_XXDS_037 e R_XXDS_039 e registro da data de execução do BUC. A tabela possui um ícone no nome da classe indicando que a organização é desordenada, não havendo chave de organização. O ícone no atributo ID_ROT_BUC indica que é índice árvore B^+ e a tabela está indexada por IX_ID_ROT_BUC( ), conforme o estereótipo <<IX>>. Foi utilizado o índice árvore B^+ para melhor desempenho no acesso aos dados.

R_XXDS_042: É uma tabela que identifica quais fitas magnéticas, dentre as 829 atuais, serão utilizadas no BUC quando houver rotina de salvamento dos dados. A quantidade de fitas aumentará com a aquisição de novas fitas ou outro meio de armazenamento dos dados. A tabela possui um ícone no nome da classe indicando que a organização é ordenada pela chave ID_ROT_BUC, conforme o estereótipo <<KY>>. O ícone no atributo ID_FITA indica índice árvore B^+ para melhor

desempenho no acesso aos dados, indexado por IX_ID_FITA( ), conforme o estereótipo <<IX>>.

R_XXDS_044: É uma tabela que mantém o histórico das fitas alocadas e que podem ser liberadas. A tabela possui um ícone no nome da classe indicando que a organização é por função pela chave NM_FITA, conforme o estereótipo <<KY>>. Não foi especificado índice para essa tabela.

R_XXDS_045: É uma tabela utilizada para a descrição dos tipos de fitas. Há somente 5 tipos e não há previsão de novos tipos. A tabela possui um ícone no nome da classe indicando que a organização é desordenada, não havendo chave de organização. Não foi especificado índice, ficando a critério do gerenciador do banco de dados. Ou seja, nesse caso, a chave primária PK_ID_ TIPO( ) é o índice primário.

R_XXDS_046: Essa tabela mantém uma cópia de segurança dos dados da tabela R_XXDS_040. A tabela possui um ícone no nome da classe indicando que a organização é por função, pela chave NM_FITA, conforme o estereótipo <<KY>>. Não foi especificado índice.

5.4 CONSIDERAÇÕES FINAIS

Até o momento não foi possível a obtenção dos resultados de performance, pois o projeto foi implementado recentemente e, não há dados suficientes para apresentar um comparativo dos ganhos de desempenho obtidos com a extensão da UML nesse projeto de banco de dados.

Por outro lado, a proposta já apresenta a necessidade de os projetistas pensarem sobre o projeto e evitarem as práticas cíclicas do uso dos mesmos índices, como o árvore B^+, que é o mais empregado. Ainda, no caso do banco de dados Oracle é possível que o uso de índice seja inicialmente legado ao

segundo plano já que o gerenciador do banco possui a característica de utilizar o método de acesso “Choose” (CBO - *Cost Based Optimizer*) ao invés do “Rule” (RBO - *Rule Based Optimizer*), o que pode ocasionar problema de desempenho no sistema. Ou seja, quando o banco de dados trabalha configurado para “Choose” ele mesmo determina os melhores métodos de acesso aos dados, baseado em estatísticas de uso das tabelas ou dos esquemas. Embora a Oracle recomende o uso do “Choose”, na prática, os analistas e os administradores do banco de dados são os profissionais que entendem do negócio e podem então determinar as melhores práticas. O emprego do /*rule*/ na programação PL/SQL obrigará o banco a utilizar o índice especificado e não as estatísticas do “Choose”.

Capítulo 6. CONCLUSÃO

6.1 CONSIDERAÇÕES INICIAIS

Motivado pela necessidade atual da engenharia de banco de dados, que visa desempenho no armazenamento e recuperação dos dados em bancos de dados cada vez maiores, o presente trabalho busca conduzir os projetistas no detalhamento e representação dos aspectos importantes do projeto físico de banco de dados com o uso da extensão do UML *profile* na modelagem dos dados.

Este capítulo finaliza a exposição com o relato das contribuições, além da enumeração e discussão de futuros trabalhos e pesquisas.

6.2 TRABALHOS FUTUROS

Nesse trabalho foi apresentada a extensão da UML com o uso do UML *profile* de banco de dados para aplicações em projetos físicos. A vantagem no uso da extensão proposta está em propiciar recursos gráficos para que definições de aspectos físicos de um banco de dados possam ser especificadas durante a engenharia do banco de dados, antes de uma implantação e dos eventuais problemas de desempenho. Desta forma, melhorando significativamente a organização dos arquivos nas tabelas e os modos de acesso aos dados nas tabelas. Também foi exposto que as configurações dos servidores podem ser detalhadas com as especificações do bloco de dados e *cache* nos diagramas de componentes. Ainda, foi ilustrado que os diagramas de seqüência quando representados com os comandos SQL para acesso às tabelas, estes contribuem para as melhores práticas no reconhecimento das operações entre a aplicação cliente e o banco de dados.

O presente trabalho incentiva o incremento de novos estereótipos e ícones nas ferramentas de modelagem na UML, assegura que novas contribuições sejam

realizadas no pacote *profile* e corrobora que o projeto físico pode ser aplicado, com maior detalhe, na engenharia de banco de dados desde que sejam mantidos os conceitos básicos da UML, sua representação e a legibilidade dos diagramas estendidos.

6.3 CONSIDERAÇÕES FINAIS

A UML tem sido amplamente reconhecida como uma linguagem de modelagem para os vários aspectos dos negócios, software e sistemas de informação. Por tratar-se de uma linguagem que apresenta extensão, possui mecanismos que possibilitam a inclusão de novos elementos para os domínios específicos, como a modelagem dos negócios, aplicações *web*, bancos de dados, desenvolvimento de software, *data warehouses*, entre outros. Embora possua elementos suficientes para a representação dos aspectos conceituais e lógicos, não detalha os aspectos de desempenho do projeto físico e durante muitos anos pouco se fez pela modelagem do projeto físico nas fases iniciais de uma engenharia de banco de dados.

O objetivo principal foi alcançado com a possibilidade de estender a UML com o uso do seu *profile* de banco de dados para o projeto físico, de forma que os projetistas utilizem essa ferramenta como apoio no desenvolvimento de projetos mais detalhados e com maior desempenho.

A proposta de extensão mostra-se promissora, considerando que as exigências de especificação para sua utilização são aquelas freqüentemente recomendadas por experientes administradores e consultores, em projetos de banco de dados, mas que são geralmente relegadas a um segundo plano (ou posterior implementação do banco de dados). O caso de estudo permitiu comprovar o valor desta extensão, não para ser um recurso opcional, mas de utilização obrigatória, principalmente em casos em que o banco de dados é um elemento estratégico para uma instituição ou organização.

REFERÊNCIAS BIBLIOGRÁFICAS

AKEHURST, D.H.; BORDBAR, B.; RODGERS, P.J.; DALGLIESH, N.T.G. “Automatic Normalisation via Metamodelling”, In Workshop on Declarative Meta Programme to Support Software Development, 2002.

ALHIR, Sinan Si. “Unified Modeling Language Extension Mechanisms”, In Distributed Computing – December 1998. Disponível em http://www.DistributedComputing.com, acessado em fevereiro/2004.

AVANZI, Edson Luiz; CAMOLESI Jr, Luiz. “Extending UML for Database Physical Design”, IRMA - International Resources Management Association, 23-26 May, 2004, LA, USA.

BAGUI, Sikha; EARP, Richard. “Database Design Using Entity-Relationship Diagrams”, Florida : Auerbach Publications, 2003.

BJÖRKANDER, Morgan; KOBRYN, Cris. “Architecting Systems with UML 2.0”, IEEE Computer Society, July/August 2003.

BOOCH, Grady; RUMBAUGH, James; JACOBSON, Ivar. “The Unified Modeling Language User Guide”, Addison-Wesley Pub Co, 1998.

BOSCH, Robert; STOLTE, Cris; TANG, Diane; GERTH, John; ROSENBLUM, Mendel; HANRAHAN, Pat. “New Visualization Techniques”, In ACM SIGGRAPH Computer Graphics Newsletter, Vol. 34, No. 1, February 2000.

CATARCI, Tiziana; COSTABILE, Maria F.; MATERA, Maristella. “Visual Metaphors for Interacting with Databases”, In ACM SIGCHI Bulletin, Vol. 27, No. 2, 1995.

CHEN, Peter. “The Entity-Relantionship Model – Toward a Unified View of Data”, In ACM, Vol. 1, No. 1, March 1976.

CHO, H.B. "IDEF Overview", Manufacturing Systems Integration Lab Department of Industrial Engineering Pohang University of Science & Technology, 2000.

CHOENNI, Sunil; BLANKEN, Henk M.; CHANG, Thiel. “On The Automation of Physical Database Design”, In ACM SAC’93/2/93/IN, USA.

CONEJERO, J.; HERNÁNDEZ, J.; PEDRERO, J. “Definición de un Perfil UML para el Aspecto de Notificatión en Entornos Distribuidos CORBA”. Disponível em http://quercusseg.unex.es/juanmamu/DSOA04/papers/conejero-hernandez-pedrero.pdf, acessado em janeiro/2006.

CODD, E. F. “The Relational Model for Database Management”, Version 2, New York : Addison-Wesley publishing, 1990.

CURTIS, Bill; KELLNER, Marc T.; OVER, Jim. "Process Modeling", Communications of the ACM, Vol. 35, No. 9, September 1992.

DATE, C. J. “The Database Relational Model : A Retrospective Review and Analysis”, Addison-Wesley, 2001.

DAVENPORT, Thomas H. “Ecologia da Informação”, tradução de Bernadette Siqueira Abrão, São Paulo : Futura, 1998.

DORSEY, Paul. “UML Class Models as a Data Modeling Tool”. Disponível em http://www.datawarehouse.com/article/?articleid=3040, acessado em fevereiro/2003.

ELMASRI, Ramez; NAVATHE, Shamkant B. "Fundamentals of Database Systems”, 3rd ed., Addison-Wesley, 2000.

FINKELSTEIN, S.; SCHKOLNICK M.; TIBERIO P. “Physical Database Design for Relational Databases”, In ACM Transactions On Database Systems (TODS), Vol. 13, No. 1, March 1988.

FOWLER, Martin. “UML Distilled : A Brief Guide To The Standard Object Modeling Language”, 3rd ed., Addison-Wesley, 2004.

FURLAN, José Davi. Modelagem de objetos através da UML – the Unified Modeling Language, São Paulo: Makron Books, 1998.

GORNIK, Davor. “The UML and Data Modeling”. Disponível em <http://www.rational.com>, acessado em agosto/2003.

IBM Corporation. “The UML and Data Modeling”. Disponível em http://www.rational.com/media/whitepapers/Tp180.PDF, acessado em novembro/2003.

IBM Corporation. “A Preview of UML 2.0”. Disponível em http:// www.rational.com, acessado em janeiro/2003.

IBM Corporation. “IBM Red Brick Warehouse 6.3”, Updated in March 2004. Disponível em http://publib.boulder.ibm.com/infocenter/rb63help/ index.jsp? topic=/ com.ibm.redbrick.doc6.3/perf/perf25.htm, acessado em novembro/2004.

IDEF. “"Integrated Definition Methods". Disponível em http://www.idef.com/, acessado em junho/2002.

IEEE Computer Society. “IEEE Standard for Conceptual Modeling Language Syntax and Semantics for IDEF1X97 (IDEF object)”, IEEE Std 1320.2-1998, December 2003.

KAMATH, Manjunath; DALAL P. Nikunj; CHAUGULE Amit; SIVARAMAN, Eswar; KOLARIK, William J. “A Review of Enterprise Process Modeling Techniques”, Oklahoma State University, 2004. Disponível em http://www.okstate.edu/cocim/members/eswar/Fall2003/SYST520/SESBookChapter2.pdf, acessado em maio/2004.

KELLNER, Mark I.; ROMBACH, H. Dieter. "Comparisons of Software Process Descriptions", In Proceedings of the Sixth International Software Process Workshop, Hakodate, Japan, October 1990, IEEE Computer Society Press, 1991.

KIVISTO, Kari. “Roles of Developers as Part of a Software Process Model”, In Proceedings of the 32nd Hawaii International Conference on System Sciences – 1999.

LARMAN, Craig. “Utilizando UML e Padrões Orientados a Objetos”, tradução de Luiz A. Meirelles Salgado, Porto Alegre : Bookman, 2000.

LEE, John Peter; GRINSTEIN, Georges G. “An Architecture for Retaining and Analyzing Visual Explorations of Databases”, In Proceedings of the 6th IEEE Visualization Conference – 1995.

LONEY, Kevin; THERIAULT, Marlene. Oracle8i O Manual do DBA, do original “Oracle8i DBA Handbook”, tradução de Kátia Roque, 1ª edição, Rio de Janeiro : Campus, 2000.

MACKINNON, Neil; MURPHY, Steve. “Designing UML Diagrams for Technical Documentation”, In Proceedings of the 21st ACM annual international conference on Documentation (SIGDOC’03).

MARCA, David A.; MCGOWAN, Clement L. “SADT: Structured Analysis and Design Techniques”, McGraw-Hill, New York : 1988.

MARCUS, Aaron. “Metaphor Design in User Interfaces: How to Manage Expectation, Surprise, Comprehension, and Delight Effectively”, In ACM Conference on Human Factors in Computing Systems (SIGCHI’97).

MARTIN, Grant. “UML for Embedded Systems Specification and Design: Motivation and Overview”, In Proceedings of the 2002 Design, Automation and Test in Europe Conference and Exhibition (DATE’2002), IEEE Computer Society.

MCFADDEN, Fred R.; HOFFER, Jeffrey A.; PRESCOTT Mary B. Modern Database Management. 5th ed., Addison-Wesley, 1999.

MELTON, Jim; SIMON, Alan R. “SQL 1999 Understanding Relational Language Concepts”, San Diego : Academic Press, 2002.

MEMON, Nasrullah. “Indexing Structures for Files”, F7S/MIT, October 04, 2004. Disponível em http://cs.aaue.dk/~nasrullah/Fall04/lecture_notes/week4_2B.pdf, acessado em novembro/2004.

MOHNKERN, Ken. “Visual Interaction Design: Beyond the Interface Metaphor”, In ACM SIGCHI Bulletin, Vol. 29, No. 2, April 1997.

MOK, Wai Yin; PAPER, David P. “On Transformations from UML Models to Object-Relational Databases”, In Proceedings of the 34th Hawaii International Conference on System Sciences – 2001.

MOLINA, Hector Garcia; ULLMAN, Jeffrey D., WIDOM, Jennifer. “Database System Implementation”, Prentice Hall, 2000.

MOORE, Jim. “Logical and Physical Index Concepts: Part I – Logical Aspects of Database Indexing”, in Technical Enterprises Inc., 1996. Disponível em http://www.naspa.com/PDF/96/T9604007.pdf, acessado em novembro/2004.

MORA, Sergio Lujan; TRUJILLO, Juan. “Physical Modeling of Data Warehouses using UML”, in Proceedings of the 7th ACM International Workshop on Data Warehousing and OLAP, 2004.

MULLER, Robert J. “Database Design for Smarties”, Morgan Kaufman, 1999.

NAIBURG, Eric J.; MAKSIMCHUK, Robert A.. “UML for database design”, 1st edition, Addison-Wesley Pub Co, 2001.

NASSAR, Mahmoud. “VUML : a Viewpoint oriented UML Extension”, in Proceedings of the 18th IEEE International Conference on Automated Software Engineering, 2003.

NORAN, Ovidiu S. “Business Modelling:UML vs IDEF”, Griffith University School of Computing and Information Technology. Disponível em http://www.cit.gu.edu.au/~noran/Docs/UMLvsIDEF.pdf, acessado em julho/2004.

OMG (Object Management Group). “UML 2.0 Infrastructure Final Adopted Specification”, OMG document ptc/03-09-15, Object Management Group, 2003.

OMG (Object Management Group). “UML 2.0 Superstructure Final Adopted Specification”, OMG document ptc/03-08-02, Object Management Group, 2003.

ORACLE, Documentation Library. "Oracle8i Adminstrator´s Guide Release 8.1.5”. Disponível em http://www.docs.cs.huji.ac.il/doc/server.815/a67772/dba.htm#4755, acessado em abril/2002.

ORACLE, Metalink. “All about Bitmap Indexes”, Doc ID 70067.1, Last Revision Date 24-JUL-2002. Disponível em: http://metalink.oracle.com/metalink/plsql/ml2_documents.showDocument?p_database_id=NOT&p_id=70067.1,acessado em outubro/2004.

ORACLE, Metalink. "eTRM Technical Reference". Disponível em: https://etrm.oracle.com/pls/trm1159/etrm_fndnav.ls_object?c_name=*&n_appid=200&c_type=FILE. Acessado em outubro/2004.

ORACLE VIEW. “A Guide to Oracle Performance Tuning, Part 1”. Disponível em http://www.oreview.com/9711harr.htm, acessado em julho/2004.

RAMAKRISHNAN, M. V. “File Organization and Indexing”. Monash University, revised on 4-3-2003. Disponível em http://www.csse.monash.edu.au/courseware/cse3000/notes/file4.pdf, acessado em agosto/2004.

RATIONAL. “The UML and Data Modeling”, November 2003, Rational Software Corporation. Disponível em http://www.rational.com/media/whitepapers/Tp180.PDF, acessado em maio/2004.

SELONEM, Petri; KOSKIMIES, Kai; SAKKINEN, Markku. “How to Make Apples from oranges in UML”, In Proceedings of the 32nd Hawaii International Conference on System Sciences - 2001.

SILBERSCHATZ, Abraham; KORTH, Henry F.; SUDARSHAN, S. “Database System Concepts”, 4th ed., New York : McGraw Hill, 2002.

SPARKS, Geoffrey. “Database Modelling in UML”, In Methods & Tools e-newletter. Disponível em http://www.martinig.ch/mt/index.html, acessado em agosto/2002.

SRIVASTAVA, Jaideep; NGO, Hung Q. “Statiscal Databases”, TR99-099, Departament of Computer Science and Engineering - University of Minnesota. Disponível em https://wwws.cs.umn.edu/tech_reports_upload/tr1999/99-009.pdf, acessado em outubro/2004.

TERRASSE, Marie-Noëlle; SAVONNET, Marinette; BECKER, George. “A UML-based Metamodeling Architecture for Database Design”, In Proceedings of the IEEE International Database Engineering and Applications Symposium (IDEAS’2001).

TILLEY, Scott; HUANG, Shihong. “A Qualitative Assessment of the Efficacy of UML Diagrams as a Form of Graphical Documentation in Aiding Program Understanding”, In Proceedings of the 21st ACM annual international conference on Documentation (SIGDOC’03).

ULLMAN, Jeffrey D.; WIDOM, Jennifer. A First Course in Database Systems. New Jersey : Prentice Hall, 1997.

ULLMAN, Jeffrey D. Principles of database and knowledge – base systems. Volume I, Stanford University : Computer Science Press, 1998.

WARD, John; GRIFFITHS, Pat. “Strategic Planning for Information Systems”, 2th ed., England : Wiley, 1999.

WHITE, Stephanie; CANTOR, Murray; FRIEDENTHAL, Sanford; KOBRYN, Cris; PURVES, Byron. “Panel: Extending UML from Software to Systems Engineering”, In Proceedings of the 10th IEEE International Conference and Workshop on the Engineering of Computer-Based Systems (ECBS’2003).

YAO, S. B. “Database Storage Structure Design”, In Proceedings of the Conference on Database Engineering (Amagi, Japan, Nov. 1979). IBM, San Jose, Calif., 1979.

APÊNDICE

R_XXDS_033

Row #	ID_RODIZIO	TX_DESC_RODIZIO	CREATED_BY	CREATION_DATE	LAST_UPDATED_BY	LAST_UPDATE_DATE
1	1	Rodízio de backup diário, semanal e mensal	1835	8-out-2004 7:46:16	1835	8-out-2004 7:46:16
2	2	Rodízio de backup mais antigo	1835	8-out-2004 7:46:16	1835	8-out-2004 7:46:16
3	3	Rodízio de backup único	1835	8-out-2004 7:46:16	1835	8-out-2004 7:46:16
4	4	Lê o backup mais recente	1835	8-out-2004 7:46:16	1835	8-out-2004 7:46:16
5	5	Sem rodízio	1835	8-out-2004 7:46:16	1835	8-out-2004 7:46:16

R_XXDS_034

Row #	ID_MENSAGEM	TX_MENSAGEM	CREATED_BY	CREATION_DATE	LAST_UPDATED_BY	LAST_UPDATE_DATE
1	16	BUC DE VERIFICAR LOG/ESPACO DOS SERVIDORES	1835	8-out-2004 7:46:09	1835	8-out-2004 7:46:09
2	9	ACERTAR TABELA KM019 C/ SEMANA REF. CARGA MAQUINA.	1835	8-out-2004 7:46:09	1835	8-out-2004 7:46:09
3	3	ROTINA DEXP (ENCER.SI) - ESPERAR USUARIO SOLICITAR.	1835	8-out-2004 7:46:09	1835	8-out-2004 7:46:09
4	4	ROTINA SAV (ENCER.SI) - ESPERAR SOLICIT. USUARIO SP.	1835	8-out-2004 7:46:09	1835	8-out-2004 7:46:09
5	6	ENCER.KE (SOLIC.USUARIO) ACERTAR CARTAO CONTR.	1835	8-out-2004 7:46:09	1835	8-out-2004 7:46:09
6	7	ACERTAR CARTAO CONTR SE O USUARIO SOLICITAR.	1835	8-out-2004 7:46:09	1835	8-out-2004 7:46:09

R_XXDS_035

Row #	ID_LOCALIZACAO	TX_LOCALIZACAO	CREATED_BY	CREATION_DATE	LAST_UPDATED_BY	LAST_UPDATE_DATE
1	1	Fita Rodízio	1835	8-out-2004 7:46:16	1835	8-out-2004 7:46:16
2	2	Fita Permanente	1835	8-out-2004 7:46:16	1835	8-out-2004 7:46:16
3	3	Fita Terceiros	1835	8-out-2004 7:46:16	1835	8-out-2004 7:46:16

4	4	Fita Trabalho	1835	8-out-2004 7:46:16	1835	8-out-2004 7:46:16
5	5	Fita Cofre	1835	8-out-2004 7:46:16	1835	8-out-2004 7:46:16
6	6	Fita Temporária	1835	8-out-2004 7:46:16	1835	8-out-2004 7:46:16

R_XXDS_036

Row #	ID_MAQUINA	NM_MAQUINA	CREATED_BY	CREATION_DATE	LAST_UPDATED_BY	LAST_UPDATE_DATE
1	1	SUN E250	1835	8-out-2004 7:46:16	1835	8-out-2004 7:46:16
2	2	SRVAPLIC	1835	8-out-2004 8:20:50	1835	8-out-2004 8:20:50
3	3	SRVORAP	1835	8-out-2004 8:20:57	1835	8-out-2004 8:20:57
4	4	SUN E3500	1835	8-out-2004 8:21:09	1835	8-out-2004 8:21:09
5	5	KRIPTON	1835	8-out-2004 8:21:16	1835	8-out-2004 8:21:16
6	6	MERCURIO	1835	8-out-2004 8:21:23	1835	8-out-2004 8:21:23

R_XXDS_037

Row #	ID_BUC	ID_RODIZIO	ID_MENSAGEM	ID_RESPONSAVEL	NM_BUC	TX_TP_BUC	TX_SOLICITACAO	TX_SUSPENSO	TX_DUPLICIDADE	TX_TIPO_EXECUCAO
1	456	2		1330	RPDBU281	Normal	N	N	N	
2	457			1330	RPDBU283	Normal	S	N	N	
3	458	1		1826	RPDAT284	Normal	S	N	N	
4	459	1		1826	RPDBU288	Normal	N	N	N	Diária
5	460	1	12	1826	RPDAT290	Normal	N	N	N	Mensal
6	461			1330	RPDBU292	Normal	S	N	N	

TX_OBSERVACAO	NR_FLAG	CREATED_BY	CREATION_DATE	LAST_UPDATED_BY	LAST_UPDATE_DATE
RPDBKPOR PORTAL SUN E3500	1	1835	8-out-2004 7:46:02	1835	8-out-2004 9:29:09
BKP SEMANAL ORACLE DES1 - SRVORAP	1	1835	8-out-2004 7:46:02	1835	8-out-2004 7:46:02
ACERTO DE DADOS NO FASTBI	3	1835	8-out-2004 7:46:02	1835	3-nov-2004 13:41:16
FBI-BKP2 (BACKUP 2 DO FAST-BI)	3	1835	8-out-2004 7:46:03	1835	9-nov-2004 7:41:24

FBI-MENS (ROTINA MENSAL NO FAST-BI)	3	1835	8-out-2004 7:46:03	1835	9-nov-2004 7:40:13
RPDBK11I BKP ESTANCIA TEST (SRVORAP)	1	1835	8-out-2004 7:46:03	1835	8-out-2004 7:46:03

R_XXDS_038

Row #	ID_PERIODICIDADE	ID_BUC	TX_SEGUNDA	TX_TERCA	TX_QUARTA	TX_QUINTA	TX_SEXTA	TX_DT_EXECUCAO	DT_INI_EFETIVIDADE
1	145	447	S	S	S	S	S		
2	146	448	N	N	N	N	S		
3	147	377	N	N	N	N	N		
4	148	449	N	N	N	N	N	9928	
5	149	450	N	N	N	N	N	9928	
6	150	451	N	N	N	N	N	9928	

DT_FIM_EFETIVIDADE	CREATED_BY	CREATION_DATE	LAST_UPDATED_BY	LAST_UPDATE_DATE
	1835	8-out-2004 7:46:06	1835	8-out-2004 7:46:06
	1835	8-out-2004 7:46:06	1835	8-out-2004 7:46:06
	1835	8-out-2004 7:46:06	1835	8-out-2004 7:46:06
	1835	8-out-2004 7:46:06	1835	8-out-2004 7:46:06
	1835	8-out-2004 7:46:06	1835	8-out-2004 7:46:06
11-out-2004 12:00:08	1835	8-out-2004 7:46:06	1835	8-out-2004 7:46:06

R_XXDS_039

Row #	ID_ROTINA	ID_MAQUINA	NR_DURACAO	TX_ROTINA	TX_OBSERVACAO	CREATED_BY	CREATION_DATE	LAST_UPDATED_BY	LAST_UPDATE_DATE
1	40	7	0	(LONG)	EXECUCAO DIARIA, PRIMEIRO BUC DA SCHEDULAGEM. FECHAR ACESSO AOS USUARIOS E FAZER ESTATISTICAS DO BANCO (IBMJ40) AVISAR IMEDIATAMENTE.	1835	8-out-2004 7:46:01	1835	9-out-2004 9:00:02
2	41	9	0	(LONG)		1835	8-out-2004 7:46:01	1835	9-out-2004 9:02:06

3	42	7	0	(LONG)	EXECUCAO SOB SOLICITACAO NO IBMJ40. EXECUTAR APOS RPDBKADA. SE CANCELAR VOLTAR COM TPDRSALL O BKP RPDBKADA. AVISAR IMEDIATAMENTE.	1835	8-out-2004 7:46:01	1835	9-out-2004 8:10:16
4	43	6	0	(LONG)	ROTINA DE ATUALIZACAO DO CEP BANCARIO. AVISAR DIA SEGUINTE.	1835	8-out-2004 7:46:01	1835	11-out-2004 12:56:24
5	44	9	0	(LONG)	EXECUTAR SOMENTE NO SABADO A TARDE. EXECUTAR SEMPRE APOS "QR.BAT - INDEXACAO" E ANTES "EXCLUI-LINUX".	1835	8-out-2004 7:46:01	1835	11-out-2004 12:33:04
6	45	7	0	(LONG)	EXECUTAR SEMPRE QUE O NUCLEO DO ADABAS ONLINE SAIR DO AR ANORMAL. AVISAR IMEDIATAMENTE.	1835	8-out-2004 7:46:01	1835	9-out-2004 8:11:14

R_XXDS_040

Row #	ID_FITA	ID_RESPONSAVEL	ID_LOCALIZACAO	ID_TIPO	ID_SISTEMA	TX_NR_FITA	NM_FITA	TX_TP_FITA	TX_DESCRICAO
1	104	1835	4	1		2117	TRABALHO	RXX	
2	105	1835	1	1		1299	TRIBKSYS	RPD	RPDBKSYS TRIMESTRAL
3	106	1835	1	1		1330	RPDBKPRT	RPD	RPDBKPRT - BKP PROTHEUS - PRODUCAO
4	107	1328	2	1		0970	RPDBKC10	RPD	RPDBKC10 - BKP FULL IBMC10
5	108	1835	1	1		1397	RPDBKEXP	RPD	RPDBKEXP - TER
6	109	1835	1	1		1322	RPDBKEXP	RPD	RPDBKEXP - QUA

DT_EXECUCAO	TX_DIA_FITA	NR_SEQUENCIA	TX_DESTINO	TX_ALTERACAO	TP_SITUACAO	DT_INICIO_ALOCACAO	DT_FIM_ALOCACAO	DT_INICIO_EFETIVIDADE
		0						
31-jul-2003 6:43:56	TRIMESTR	0						
15-out-2004 10:07:36	SEM01	0	OPERACAO	S				
26-mar-1999 6:11:12	SEMANAL	0	OPERACAO					
9-nov-2004 6:33:30	TER	0		S				
3-nov-2004 5:48:46	QUA	0		S				

DT_FIM_EFETIVIDADE	CREATED_BY	CREATION_DATE	LAST_UPDATED_BY	LAST_UPDATE_DATE
	1835	8-out-2004 7:46:13	1835	8-out-2004 7:46:13

	1835	8-out-2004 7:46:13	1835	8-out-2004 7:46:13
	1835	8-out-2004 7:46:13	1835	8-out-2004 7:46:13
31-dez-1999	1835	8-out-2004 7:46:13	1835	8-out-2004 7:46:13
	1835	8-out-2004 7:46:13	1835	8-out-2004 7:46:13
	1835	8-out-2004 7:46:13	1835	8-out-2004 7:46:13

R_XXDS_041

Row #	ID_ROTINA	ID_BUC	ID_ROT_BUC	DT_EXECUCAO	CREATED_BY	CREATION_DATE	LAST_UPDATED_BY	LAST_UPDATE_DATE
1	3	332	3		1835	8-out-2004 7:46:01	1835	8-out-2004 7:46:01
2	4	333	4	29-out-2004 10:15:56	1835	8-out-2004 7:46:01	1835	8-out-2004 7:46:01
3	5	334	5		1835	8-out-2004 7:46:01	1835	8-out-2004 7:46:01
4	6	335	6	9-nov-2004 6:33:30	1835	8-out-2004 7:46:01	1835	8-out-2004 7:46:01
5	7	336	7		1835	8-out-2004 7:46:01	1835	8-out-2004 7:46:01
6	8	337	8		1835	8-out-2004 7:46:01	1835	8-out-2004 7:46:01

R_XXDS_042

Row #	ID_FITA	ID_ROT_BUC	NM_FITA	NM_FITA2	NM_FITA3	NM_FITA4	NM_FITA5	NM_FITA6	NM_FITA7	NM_FITA8	TX_PERIODO	CREATED_BY	CREATION_DATE
10		11										1835	8-out-2004 7:46:01
11		12										1835	8-out-2004 7:46:01
12		13										1835	8-out-2004 7:46:01
13		14										1835	8-out-2004 7:46:01
14		15	RPDBKSCH									1835	8-out-2004 7:46:01
15		16	RPDBKSC2									1835	8-out-2004 7:46:01

LAST_UPDATED_BY	LAST_UPDATE_DATE
1835	8-out-2004 7:46:01
1835	8-out-2004 7:46:01
1835	8-out-2004 7:46:01

1835	8-out-2004 7:46:01
1835	8-out-2004 7:46:01
1835	8-out-2004 7:46:01

R_XXDS_044

Row #	ID_FITA	ID_FITA_LIBERADA	ID_RESPONSAVEL	ID_LOCALIZACAO	NM_FITA	TX_NR_FITA	TX_DESCRICAO	DT_GRAVACAO
1	802	802	1484	6	GAMA11I	564	BKP /GAMA11I/APP - SRVORAP	21-jul-2004 0:12:00
2	304	304	1484	6	BKSISPRO	2690	BKP SRVSISPRO RECURSO SISPRO - PASTA SISPRO	12-dez-2002 15:00:00
3	298	298	1484	6	RPDBK250	2675	RPDBK250 - SISTEMA OPERACIONAL DO E250	6-nov-2003 14:50:00
4	689	689	1484	6	RPDBKOET	469	RPDBKOET - EMAIL DO E250	6-nov-2003 15:07:00
5	679	679	1484	6	RPDBKSTA	119	RPDBKSTA - /STAGE11I - SRVORAP	27-fev-2004 18:04:00
6	395	395	1484	6	WPDBKAPR	141	WPDBKAPR /PROD11I - DLT4000	27-jul-2004 15:40:00

DT_FIM_EFETIVIDADE	DT_INICIO_ALOCACAO	DT_FIM_ALOCACAO	CREATED_BY	CREATION_DATE	LAST_UPDATED_BY	LAST_UPDATE_DATE
		15-out-2004 12:15:19	1484	15-out-2004 11:43:05	1484	15-out-2004 11:43:05
		15-out-2004 12:14:14	1484	15-out-2004 11:43:02	1484	15-out-2004 11:43:02
		15-out-2004 12:15:19	1484	15-out-2004 11:43:07	1484	15-out-2004 11:43:07
		15-out-2004 12:16:19	1484	15-out-2004 11:43:10	1484	15-out-2004 11:43:10
		15-out-2004 12:16:19	1484	15-out-2004 11:43:12	1484	15-out-2004 11:43:12
		15-out-2004 12:16:19	1484	15-out-2004 11:43:14	1484	15-out-2004 11:43:14

R_XXDS_045

Row #	ID_TIPO	TX_TIPO	CREATED_BY	CREATION_DATE	LAST_UPDATED_BY	LAST_UPDATE_DATE
1	2	DDS2	1835	8-out-2004 7:46:16	1835	8-out-2004 7:46:16
2	3	DDS3	1835	8-out-2004 7:46:16	1835	8-out-2004 7:46:16
3	4	DLT	1835	8-out-2004 7:46:16	1835	8-out-2004 7:46:16
4	5	Disquete	1835	8-out-2004 7:46:16	1835	8-out-2004 7:46:16
5	1	DDS1	1835	8-out-2004 7:46:16	1835	8-out-2004 7:46:16

R_XXDS_046

Row #	ID_FITA	ID_RESPONSAVEL	ID_LOCALIZACAO	ID_SISTEMA	ID_TIPO	TX_NR_FITA	NM_FITA	TX_TP_FITA	TX_DESCRICAO
1	523	1328	2			0332	IN68-97	RIN	IN68-97 (BKP MENSAL DE 06/98 ATE 07/98)
2	521	1328	2			0330	IN68-97	RIN	IN68-97 (PRIMEIRO BKP SISTEMA ASM COMPLETO)
3	522	1328	2			0331	IN68-97	RIN	IN68-97 (BKP MENSAL ASM COMPLETO)
4	524	1328	2			0334	IN68-97	RIN	IN68-97 (BKP MENSAL DE 11/97 ATE 06/98)
5	525	1328	2			0335	IN68-97	RIN	IN68-97 (BKP SEMANAL DE 12/97 ATE 08/98)
6	526	1328	2			0336	IN68-97	RIN	IN68 - DE 07 ATE 11 DE 1997

NR_SEQUENCIA	TX_DESTINO	DT_EXECUCAO	TX_ALTERACAO	TX_DIA_FITA	DT_FIM_EFETIVIDADE	CREATED_BY	CREATION_DATE	LAST_UPDATED_BY	LAST_UPDATE_DATE
0	OPERACAO	1-jun-1998 12:00:00				1835	8-out-2004 7:46:14	1835	8-out-2004 7:46:14
0	OPERACAO	1-jul-1997 12:00:00				1835	8-out-2004 7:46:14	1835	8-out-2004 7:46:14
0	OPERACAO	8-set-1997 9:14:00				1835	8-out-2004 7:46:14	1835	8-out-2004 7:46:14
0	OPERACAO	10-nov-1997 12:00:00				1835	8-out-2004 7:46:14	1835	8-out-2004 7:46:14
0	OPERACAO	10-dez-1997 12:00:00				1835	8-out-2004 7:46:14	1835	8-out-2004 7:46:14
0	OPERACAO	25-jul-1997 12:00:00				1835	8-out-2004 7:46:14	1835	8-out-2004 7:46:14