UNIVERSIDADE REGIONAL DE BLUMENAU

CENTRO DE CIÊNCIAS EXATAS E NATURAIS

CURSO DE CIÊNCIAS DA COMPUTAÇÃO

(Bacharelado)

ANÁLISE COMPARATIVA ENTRE AMBIENTES ORACLE RELACIONAL VERSÃO 7 E ORACLE OBJETO RELACIONAL VERSÃO 8, UTILIZANDO PADRÕES DA NORMA ISO/IEC 9126.

TRABALHO DE CONCLUSÃO DE CURSO SUBMETIDO À UNIVERSIDADE REGIONAL DE BLUMENAU PARA A OBTENÇÃO DOS CRÉDITOS NA DISCIPLINA COM NOME EQUIVALENTE NO CURSO DE CIÊNCIAS DA COMPUTAÇÃO — BACHARELADO

DEMÉTRIUS DOMINGOS WOLFF DA SILVA

BLUMENAU, NOVEMBRO/1999

1999/2-08

ANÁLISE COMPARATIVA ENTRE AMBIENTES ORACLE RELACIONAL VERSÃO 7 E ORACLE OBJETO RELACIONAL VERSÃO 8, UTILIZANDO PADRÕES DA NORMA ISO/IEC 9126.

DEMÉTRIUS DOMINGOS WOLFF DA SILVA

ESTE TRABALHO DE CONCLUSÃO DE CURSO, FOI JULGADO ADEQUADO PARA OBTENÇÃO DOS CRÉDITOS NA DISCIPLINA DE TRABALHO DE CONCLUSÃO DE CURSO OBRIGATÓRIA PARA OBTENÇÃO DO TÍTULO DE:

BACHAREL EM CIÊNCIAS DA COMPUTAÇÃO

_____

Prof. Oscar Dalfovo - Orientador na FURB

_____

Prof. José Roque Voltolini da Silva - Coordenador do TCC

BANCA EXAMINADORA

_____

Prof. Oscar Dalfovo

_____

Prof. Marcel Hugo

_____

Prof. Wilson Pedro Carli

AGRADECIMENTOS

À minha esposa Cleide,

à minha família,

aos meus amigos,

à cidade de Lages, minha terra natal,

ao meu Orientador Prof. Oscar Dalfovo,

aos professores do curso de Ciências da Computação da FURB,

aos colaboradores da HBTec – Heller Brasil Tecnologia,

a todos que colaboraram direta ou indiretamente com este trabalho e

em especial, a Deus,

o meu muito obrigado.

SUMÁRIO

SUMÁRIO iv

LISTA DE FIGURAS i

LISTA DE QUADROS iii

RESUMO iv

ABSTRACT v

1 INTRODUÇÃO 1

1.1 MOTIVAÇÃO 2

1.2 ÁREA 3

1.3 JUSTIFICATIVAS / TRABALHOS CORRELATOS 3

1.4 OBJETIVOS DO TRABALHO 4

1.5 SINOPSE 4

2 FUNDAMENTAÇÃO TEÓRICA 6

2.1 DADOS 6

2.2 INFORMAÇÃO 6

2.3 EVOLUÇÃO 9

2.3.1 ARQUIVO 9

2.3.2 BANCO DE DADOS 9

2.3.3 SISTEMA DE GERENCIAMENTO DE BANCO DE DADOS 12

2.3.3.1 TRANSAÇÃO 14

2.3.3.2 ESTADOS DE TRANSAÇÕES 14

2.3.4 USUÁRIO DE BANCO DE DADOS 14

2.3.4.1 LINGUAGEM DE DEFINIÇÃO DE DADOS - DDL 16

2.3.4.2 LINGUAGEM DE MANIPULAÇÃO DE DADOS - DML 16
2.3.5 MODELAGEM DE DADOS 17
2.3.5.1 MODELOS LÓGICOS 17
2.3.5.2 MODELOS LÓGICOS HIERÁRQUICO 17
2.3.5.3 MODELOS LÓGICOS DE REDE 17
2.3.5.4 MODELO LÓGICO RELACIONAL 17
2.3.5.5 MODELO RELACIONAL 18
2.3.5.6 MODELO ENTIDADE-RELACIONAMENTO 19
2.3.5.7 MODELO ORIENTADO A OBJETO 19
2.3.6 MAPEAMENTO MODELO OBJETO PARA RELACIONAL 21
2.3.7 UML – LINGUAGEM DE MODELAGEM UNIFICADA 22
2.3.7.1 OBJETIVOS DA UML 23
2.3.8 FASES DO DESENVOLVIMENTO EM UML 23
2.3.9 COMPOSIÇÃO DA UML 25
2.3.10 NOTAÇÃO DA UML 26
2.4 BANCOS DE DADOS RELACIONAIS EXISTENTES NO MERCADO 27
2.4.1 ADABAS 27
2.4.2 ACCESS 27
2.4.3 DATAFLEX 27
2.4.4 PROGRESS 28
2.4.5 SYBASE 28
2.4.6 ORACLE 28
3 QUALIDADE DE SOFTWARE 30
3.1 HISTÓRICO DA QUALIDADE DE SOFTWARE 30

3.2 CONTROLE DA QUALIDADE DE SOFTWARE 31
3.3 NORMAS DE GARANTIA DA QUALIDADE DE SOFTWARE 32
3.4 QUALIDADE DE PRODUTOS DE SOFTWARE - ISO/IEC 9126 32
3.5 DESCRIÇÃO DA NORMA ISO/IEC 9126 32
3.5.1 CARACTERÍSTICAS E SUBCARACTERÍSTICAS DA QUALIDADE 33
3.5.2 MÉTRICAS DE SOFTWARE 34
3.5.2.1 MÉTRICAS EXTERNAS 34
3.5.2.2 MÉTRICAS INATERNAS 34
4 TÉCNICAS E FERRAMENTAS UTILIZADAS 36
4.1 MÉTODOS NO DESENVOLVIMENTO DE SISTEMAS 36
4.2 ANALISE ESTRUTURADA DE SISTEMAS 37
4.2.1 CICLO DE VIDA DE UM PROJETO ESTRUTURADO DE SISTEMA 37
4.2.2 FERRAMENTAS DE UMA ANÁLISE ESTRUTURADA 38
4.2.2.1 DICIONÁRIO DE DADOS 38
4.3 DIAGRAMA DE CONTEXTO 38
4.4 DIAGRAMA DE FLUXO DE DADOS – D.F.D. 39
4.5 MODELO ENTIDADE X RELACIONAMENTO – M.E.R. 40
4.6 TERMINOLOGIA BÁSICA DO MER 40
4.7 DIAGRAMA ENTIDADE RELACIONAMENTOo – D.E.R. 41
4.7.1 NORMALIZAÇÃO 42
4.7.1.1 ROTEIRO DE NORMALIZAÇÃO 42
4.8 ANÁLISE ORIENTADA A OBJETOS 44
4.8.1 METODOLOGIA PARA ANÁLISE ORIENTADA A OBJETOS 44
4.8.1.1 MODELO OBJETO RELACIONAL 45

4.9 FERRAMENTAS DE ANÁLISE ORIENTADO A OBJETO 46
4.9.1 DICIONÁRIO DE DADOS 46
4.9.2 DIAGRAMA DE CLASSE 46
4.9.3 DIAGRAMA DE CASO DE USO 47
4.9.4 DIAGRAMA DE INTERAÇÃO 47
4.9.5 DIAGRAMA DE ESTADO 47
4.9.6 DIAGRAMA DE IMPLEMENTAÇÃO 47
4.10 LINGUAGEM DE QUARTA GERAÇÃO - SQL 48
4.11 FERRAMENTAS CASE ORACLE 48
4.11.1 CASE*METHOD 49
4.11.1.1 CASE*DICTIONARY 49
4.11.1.2 CASE*DESIGNER 50
4.11.1.3 CASE*GENERATOR 50
4.11.1.4 CASE*EXCHANGE 51
4.11.2 ORACLE FORMS 51
4.11.3 ORACLE REPORTSs 51
4.11.4 LINGUAGEM DE PROGRAMAÇÃO ESTRUTURADA PL/SQL 52
5 DESENVOLVIMENTO DO TRABALHO 53
5.1 ESPECIFICAÇÃO 53
5.2 APRESENTAÇÃO DA ESPECIFICAÇÃO 53
5.2.1 ANÁLISE DE REQUISITOS 54
5.2.2 DIAGRAMA DE CLASSES 54
5.2.3 DIAGRAMA DE SEQÜÊNCIA 56
5.2.4 DESIGN 56

5.2.5 DESCRIÇÃO DAS TABELAS 59
5.2.6 DIAGRAMA DE FLUXO DE DADOS 62
5.2.7 ATRIBUIÇÃO DE VALORES PARA ANÁLISE COMPARATIVA 63
5.2.8 CENÁRIO 63
5.3 IMPLEMENTAÇÃO 64
5.3.1 ANÁLISE COMPARATIVA E INTERPRETAÇÃO DOS RESULTADOS 67
6 CONCLUSÕES 72
6.1 DIFICULDADES DURANTE ESSE TRABALHO 72
6.2 SUGESTÕES 73
7 ANEXOS 74
7.1 ROTINAS DO PROTÓTIPO 74
7.1.1 ROTINA DE PESQUISA DE CLIENTES 74
7.1.2 ROTINA DE GERAÇÃO DAS DUPLICATAS 74
7.2 RESULTADOS DAS EXECUÇÕES 76
REFERÊNCIAS BIBLIOGRÁFICAS 78

LISTA DE FIGURAS

Figura 1: Integrantes de Sistemas de Informação.....7

Figura 2: Exemplo de diagrama de “use-case”.....26

Figura 3: Elementos gráficos do D.F.D., notação Yourdon. 39

Figura 4: Exemplo de D.F.D. 39

Figura 5: Exemplo de M.E.R.....41

Figura 6: Exemplo de D.E.R. 42

Figura 7: Comparativo entre análise de dados e análise de objetos 44

Figura 8: Posicionamento relativo dos modelos convencionais e estendidos. 46

Figura 9: Ferramenta CASE Oracle Designer 2000 53

Figura 10: Diagrama de *use-case*. Fase de análise de requisitos.....54

Figura 11: Diagrama de classes. Fase de análise.....55

Figura 12: Diagrama de seqüência. 56

Figura 13: Fase de *Design*. Definição dos pacotes.....56

Figura 14: Desenho dos objetos na ferramenta CASE Oracle.....57

Figura 15: M.E.R. do protótipo. 58

Figura 16: D.F.D. do protótipo.....62

Figura 17: Tela inicial do protótipo.....64

Figura 18: Tela de conexão. 65

Figura 19: Tela de cadastro. 65

Figura 20: Tela rotina de geração de informações. 66

Figura 21: Tela de Logs de execuções das rotinas. 66

Figura 22: Característica funcionalidade da norma ISO/IEC 9126.....67

Figura 23: Característica confiabilidade da norma ISO/IEC 9126.....68

Figura 24: Característica usabilidade da norma ISO/IEC 9126. 68

Figura 25: Característica eficiência da norma ISO/IEC 9126. 69

Figura 26: Característica manutenibilidade da norma ISO/IEC 9126. 70

Figura 27: Característica portabilidade da norma ISO/IEC 9126. 71

LISTA DE QUADROS

Quadro 1: Relação (denominação da tabela): Aluno 18

Quadro 2: Equivalência entre termos relacionais e orientado a objeto 22

Quadro 3: Histórico da qualidade de software 31

Quadro 4: Características e subcaracterísticas da norma ISO/IEC 9126..... 33

Quadro 5: Descrição das tabelas do protótipo. 59

RESUMO

Este trabalho visa o estudo dos ambientes Oracle relacional versão 7 e Oracle objeto relacional versão 8 realizando análise comparativa através dos padrões da norma ISO/IEC 9126. Para o desenvolvimento de um protótipo de Rotinas de Notas Fiscais do Sistema de Faturamento será utilizado a linguagem de modelagem de objetos unificada - UML *(Unified Modeling Language)* e análise estruturada. Ao final desse trabalho, o protótipo implementado será utilizado para a realização da análise comparativa, tendo por finalidade demonstrar características dos dois ambientes comparados, gerando assim um maior conhecimento dos ambientes Oracle relacional versão 7 e Oracle objeto relacional versão 8.

ABSTRACT

This work aims the study of the environments relational Oracle version 7 and object object Oracle version 8, examining a comparative analysis up to ISO/IEC 9126 standard. To develop a prototype of Recipiet Module of the Sales System, will be applied the Unified Modeling Language (UML) and structured analysis. The implemented prototype will be used to make the comparative analysis, with the aim of showing characteristics of both environments, resulting in a greater knowledge of the environments relational Oracle version 7 and relational object Oracle version 8.

1 INTRODUÇÃO

No início, os sistemas informatizados em sua concepção tradicional eram implementados pelo binômio programas-arquivos. Nesse contexto era indiferente se o arquivo de entrada era uma massa de cartões perfurados ou se eram linhas entradas em um terminal, ou ainda, se o arquivo de saída eram trilhas gravadas em um disco, ou linhas impressas em um formulário. Freqüentemente, após algum tempo de vida útil, tais sistemas necessitaram de uma completa reestruturação. Novas aplicações foram desenvolvidas e novos dados incluídos [OLI92].

O desenvolvimento de software tradicional é considerado, segundo [ARA95], lento, rígido e de um modo geral seus procedimentos não funcionam adequadamente. As soluções existentes requerem novos aplicativos, novos procedimentos e novas estruturas gerenciais, muitas vezes, entretanto, em choque com a própria cultura da organização. Através desses problemas, os aplicativos em geral evoluíram surgindo a reutilização de software.

Segundo [MCC93], reutilização de software refere-se ao reaproveitamento de certos componentes que já foram construídos e estão disponíveis para novas aplicações. Trata-se do reaproveitamento de código do programa, especificações do projeto, planos, documentação, conhecimento e qualquer outra informação utilizada para criar um software, levando-se em consideração a qualidade final do mesmo.

Qualidade de Software é um tema que atualmente está em evidência devido a preocupação com os altos custos na manutenção de software e a baixa conformidade nos requisitos dos mesmos. Para isso a Organização Internacional de Padrões – ISO publicou a norma que representa a atual padronização mundial para qualidade de produtos de software denominada ISO/IEC 9126.

A análise comparativa entre os ambientes banco de dados Oracle relacional 7 e Oracle objeto relacional 8, se efetivará pelo conjunto de características da norma ISO/IEC 9126, as quais permitem a visualização da qualidade de produtos de software, entre eles funcionalidade, confiabilidade, usabilidade e eficiência.

O ambiente Oracle é um conjunto de ferramentas voltadas para o desenvolvimento de sistemas abrangendo todo o ciclo de vida dos sistemas. Durante o desenvolvimento de um sistema existem as fases de planejamento, análise, projeto e implementação. Isto pode variar conforme a metodologia empregada, e, no caso do ambiente Oracle é incorporado todas estas fases tanto no modelo de dados relacional como no recente modelo objeto relacional.

Os sistemas de gerenciamento de banco de dados evoluíram iniciando dos modelos hierárquicos para os modelos de rede indo para o relacional. O modelo mais aceito atualmente no mercado é modelo relacional. O ambiente Oracle, na versão 8, deu um grande passo na tecnologia de gerenciamento de banco de dados com a introdução de um modelo objeto relacional, o qual possibilita o armazenamento de modelos complexos de negócios em um banco de dados relacional [RAM99].

Derivada dos conceitos da programação e do projeto orientado à objeto, a análise orientado a objeto é a mais recente das abordagens de desenvolvimento de sistemas. É uma nova maneira de visualizar os problemas utilizando modelos organizados a partir de conceitos do mundo real. O componente fundamental é o objeto que combina estrutura e comportamento em uma única entidade [FUR98].

Desta forma, utilizando-se dos padrões da ISO/IEC 9126, pode-se comparar o ambiente Oracle relacional versão 7 com o Oracle objeto relacional 8. Nesse contexto mostra-se oportuno na metodologia de desenvolvimento de sistemas, utilizar a análise baseada em objetos, mais especificamente a UML. Para efetuar a comparação entre os ambientes, será desenvolvido um protótipo de aplicativo na área financeira que conterá informações relativas a um sistema de faturamento, o qual será aplicado estes padrões.

1.1 MOTIVAÇÃO

A avaliação de produtos de software tem sido introduzida nas empresas que produzem ou adquirem softwares, para que seja obtido maior qualidade nos mesmos. Para que essa avaliação seja efetivada é necessário a utilização de normas que estabeleçam e avaliem requisitos de qualidade, e o processo de avaliação seja definido e estruturado.

Normalização é o processo de aplicar regras estabelecidas e executar uma atividade de maneira ordenada. Onde a utilização de normas no desenvolvimento e avaliação de software proporciona benefícios como a redução de custos e padronizações.

A norma ISO/IEC 9126 propõem um modelo de qualidade e métricas para serem utilizadas por empresas que pretendem avaliar produtos de software.

A empresa Oracle Corporation, lançou por volta de 1996, uma nova versão de seu banco de dados, denominada objeto relacional versão 8, segundo a própria empresa, implementaram-se muitas melhorias perante os ambientes anteriores, dentre elas encontram-se as melhorias no desempenho das transações, melhorias no desempenho do gerenciador de recuperação entre outros.

1.2 ÁREA

Os interessados nesse trabalho são os desenvolvedores de softwares que pretendem utilizar avaliação de produtos para aprimorar o processo de desenvolvimento e melhorar a qualidade do produto final de software desenvolvidos através dos produtos da empresa Oracle Corporation.

As pessoas ou empresas que queiram adquirir produtos desenvolvidos/criados em ambientes Oracle, e pretendem obter qualidade nos produtos adquiridos.

Professores e estudantes da disciplina de Banco de Dados, Análise de Sistemas, Sistemas de Informação e Engenharia de Software que tenham interesse em qualidade de software.

1.3 JUSTIFICATIVAS / TRABALHOS CORRELATOS

Diversos trabalhos de conclusões de cursos abordaram esse assunto, dentre eles a análise comparativa entre bancos de dados Adabas, Dataflex, Oracle e Progress realizada por [TIR98], o qual realizou análise comparativa entre bancos de dados de fabricantes diferentes utilizando as versões mais recentes de bancos de dados no mercado na época do desenvolvimento do trabalho.

Sabendo-se que atualmente uma grande quantidade de empresas adquiriram o banco de dados de propriedade da empresa Oracle Corporation versão relacional 7 e dos quais muitos pretendem migrar para a versão mais recente, objeto relacional 8, interessou-me a questão do estudo da qualidade das duas versões antes citadas.

Ainda referente ao trabalho do [TIR98], o mesmo sugestiona no término de seu trabalho a continuidade das análises comparativa em novos produtos, comparando versões anteriores com versões recentes.

1.4 OBJETIVOS DO TRABALHO

Esse trabalho de conclusão de curso visa analisar, especificar e comparar uma aplicação nos ambientes Oracle versão relacional 7 e objeto relacional 8 através dos padrões da norma ISO/IEC 9126.

1.5 SINOPSE

O trabalho foi dividido em sete capítulos conforme descrição abaixo:

O primeiro capítulo define os objetivos do trabalho, apresentando a motivação, área e justificativa e estrutura para sua elaboração.

O segundo capítulo refere-se a fundamentação teórica, descrevendo a evolução de dados até banco de dados incluindo evolução de técnicas de modelagem de dados à UML (linguagem de modelagem unificada).

O terceiro capítulo refere-se qualidade de software, descrevendo as características da norma ISO/IEC 9126.

O quarto capítulo apresenta as técnicas e ferramentas utilizadas para elaboração da especificação e implementação do protótipo.

O quinto capítulo apresenta o protótipo, sua especificação, características, telas e operacionalização.

O sexto capítulo apresenta as conclusões, dificuldades encontradas e sugestões para próximos trabalhos.

O sétimo capítulo completa o trabalho demonstrando os rotinas desenvolvidas no protótipo.

2 FUNDAMENTAÇÃO TEÓRICA

2.1 DADOS

De acordo com [COR99], os dados de uma organização são tão importantes quanto a edificação, equipamentos e patrimônio financeiro. A obtenção de dados é demorada e cara, e sua utilidade vai além das operações. O valor potencial dos recursos dos dados organizacionais é tão importante que muitas empresas estabelecem departamentos de administração de dados, e esses tem como finalidade guardar e proteger os dados, e assegurar que os mesmos sejam usados com eficiência.

Segundo[DAT91], o constante crescimento no uso de sistemas de informatizados em rotinas de diversos ramos de atividades gerou grandes volumes de dados. Ao final dos anos 50 utilizou-se do uso de armazenamento de dados em arquivos de computadores, conforme necessidade, os arquivos eram criados sem muito controle. O problema dava-se que diferentes setores dentro de uma mesma organização, desenvolviam seus próprios sistemas, gerando em muitos casos, arquivos com dados redundantes.

Verificando esse problema, veio a necessidade de um controle dos dados e sistemas, para uma maior integridade, melhor padronização nos formatos de arquivos, pois as organizações concluíram que os dados gerados adequadamente geram informações corretas e de grande valia. As dificuldades no acesso das informações mais comuns eram as seguintes: vários usuários acessando os mesmos registros, inconsistência, redundância de dados segurança e falta de integridade.

De acordo com [DAL98], desde o início da história de sistemas informatizados, verificou-se a importância no valor dos dados e no armazenamento dos mesmos em um único local, havendo um gerenciamento dos dados para evitar problemas com as informações, dentre elas, redundância, inconsistência, acesso de registros, e outros.

2.2 INFORMAÇÃO

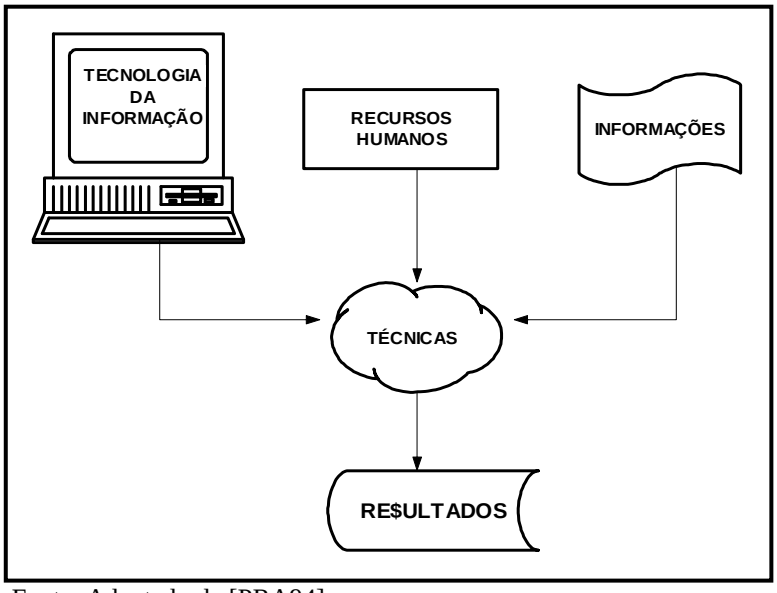
Para [OLI92], informação é o dado trabalhado o qual permite-se tomar decisões. É um recurso vital para qualquer organização que devidamente estruturado, integra os diversos

setores que a compõem. A utilização de um Sistema de Informação pode vir a facilitar processos decisórios na obtenção de dados estrategicamente escolhidos e de conteúdos relevantes para qualquer nível e tamanho de organizações.

Os Sistemas de Informação surgiram como uma forma de manter o executivo preparado, com visão integrada de todas as áreas da empresa, isso sem gastar muito tempo ou requerer do profissional um conhecimento aprofundado de cada área. Sistemas de Informação são sistemas que permitem a coleta, o armazenamento, o processamento, a recuperação e a disseminação de informações. Sistemas de Informação são, hoje, quase sem exceção baseados no computador e apoiam as funções operacionais, gerenciais e de tomada de decisão existentes nas organizações. Os usuários de Sistemas de Informação são provenientes tanto do alto nível operacional quanto do nível tático (e mesmo estratégico) e utilizam Sistemas de Informação para alcançar os objetivos e as metas de suas áreas funcionais [CHA99].

De acordo com [PRA94], Sistemas de Informação são formados pela combinação estruturada de vários elementos, organizados da melhor maneira possível, visando atingir os objetivos da organização.

Figura 1: Integrantes de Sistemas de Informação



Fonte: Adaptado de [PRA94]

Conforme figura 1, são integrantes dos Sistemas de Informação: a informação (dados formatados, textos livres, imagens e sons), os recursos humanos (pessoas que coletam, disseminam e utilizam as informações), as tecnologias de informação (o hardware e o

software usados no suporte aos sistemas de informação) e as práticas de trabalhos (métodos utilizados pelas pessoas no desempenho de suas atividades).

Segundo [OLI92], a evolução da informática em uma organização ocorre em seis estágios que refletem o posicionamento do usuário perante novos paradigmas:

a) primeiramente, tem-se a Iniciação. Nesse estágio, o usuário é resistente ao uso da informática e seu envolvimento com a tecnologia é superficial. A organização encoraja o uso da informática e se preocupa com o aprendizado, mas poucas atividades são informatizadas;

b) o segundo estágio é o Contágio, onde começam a proliferar sistemas informatizados, que automatizam atividades antes desenvolvidas manualmente, porém sem a preocupação com a integração das informações;

c) o controle é o terceiro estágio. O crescimento do uso de sistemas de informação passa a ser explosivo, o usuário sendo a força propulsora. Por isso, a organização passa a exigir melhor gestão dos recursos de informática;

d) como quarto estágio, tem-se a integração, ocorrendo em resposta à pressão por melhor gestão. Os sistemas de informação passam a ser orientados para atender às necessidades dos níveis gerenciais, as informações são de melhor qualidade e é exigida maior integração entre elas;

e) o quinto estágio é a Administração de Dados. Nesse estágio, os sistemas de informação começam a ser organizados em termos de sistema que interessam à organização como um todo e sistema de uso setorial ou especializado, sendo preciso muito cuidado, em qualquer hipótese, com a correta administração dos dados, de modo a evitar possíveis e indesejáveis redundâncias;

f) maturidade é o último estágio da evolução da informática em uma organização. A informação passa, a partir daí, a ser considerada patrimônio da organização, o usuário é participativo e responsável, e o crescimento da informática é ordenado.

2.3 EVOLUÇÃO DA INFORMÁTICA

2.3.1 ARQUIVO

O conjunto de dados fornece a base de sustentação das informações. Cada conjunto de informações, constitui um campo do registro e o relacionamento entre esses campos constitui o registro. Registro é um conjunto de informações logicamente relacionadas. Arquivo é um conjunto de registros os quais encontram-se armazenados em um dispositivo de memória secundária [FAR89].

As principais operações que podem ser realizadas em um arquivo são: consulta de um registro, inclusão de um novo registro, alteração ou exclusão de um registro no arquivo. Os registros ficam armazenados de forma a favorecer determinadas ações em comparações a outras. Existem, basicamente, dois modos de organizações de arquivos: o primeiro é a organização seqüencial, o qual os registros são armazenados em ordem de seqüência de inclusão. O segundo é a organização direta, o qual o acesso é feito de modo aleatório. Existem outras organizações, que são variações dessas duas, as quais são importantes de acordo com as aplicações [FAR89].

2.3.2 BANCO DE DADOS

De acordo com [CHU83], a idéia mais comum, tanto para a pessoa leiga em informática como para os técnicos não envolvidos em ambientes de banco de dados, é que o sistema de banco de dados numa organização corresponde a uma junção de arquivos de dados de toda a empresa em algum tipo de armazenamento magnético, sendo manipulado por um conjunto de aplicativos/programas. Entende-se por programas um conjunto de instruções logicamente ordenados e por aplicativo um conjunto de programas que atendam a uma especificação básica de informação. Tais aplicativos/programas efetuariam operações de manutenção do banco de dados, como adições, exclusões e atualizações de dados, assim como cálculos e regravação de informações mais elaboradas no banco de dados e também operações de pesquisas de informações mais difíceis para os níveis gerenciais de controle. Quando solicitado, o sistema de banco de dados forneceria informações para análises dentro de um esquema de simulação para o fornecimento de alternativas no processo de planejamento

empresarial, isto é, possibilidade de construir modelos com base nas informações do banco de dados.

Fazendo uma analogia, um arquivo convencional normalmente constitui-se por um conjunto de registros lineares que não possuem informações de relacionamento entre os mesmos. O conjunto de dados estruturado constitui um modelo simbólico de dados do conjunto de informações que representa de forma simbólica e descritiva do sistema físico. Assim, o banco de dados conterá não somente os dados essenciais e de interesse do usuário (principais), mas também dados de relação (secundários) entre os dados principais.

Tendo como finalidade o auxilio da administração do banco de dados e em função desta, uma série de outros dados secundários é considerada.

Classificação dos dados no banco de dados:

a) dados principais;

b) dados de estrutura;

c) dados como índice;

d) dados para controle de segurança e privacidade;

e) dados de dicionário de dados.

Os tipos de dados principais, de estruturas e de índice são os mais importantes e diferenciais dos arquivos convencionais. Os demais dados variam conforme a complexidade dos Sistemas de Gerenciamento de Banco de Dados (SGBD). Os dados como índices consultam com maior rapidez os dados do banco de dados, e, normalmente, provêm da escolha conveniente de alguns dos itens de dados principais.

Outra característica importante do banco de dados dá-se na sua utilização por diversas aplicações diferentes, com usuários diferentes. Mas, não é uma característica identificadora de um banco de dados.

Pela sua capacidade de interligação, entre grupos de informações, através de dados de relação, pode-se construir um grande volume de informações envolvendo diversos setores funcionais de uma empresa. Esta possibilidade de integração de dados, entre diversas áreas funcionais da empresa, produz uma redução da redundância de dados, muito freqüente nas aplicações que não utilizam banco de dados.

A idéia bastante difundida de um banco de dados é a que contenha todos os dados da empresa, mas também não é característica identificadora necessária de banco de dados. Existem muitas outras razões para que tal banco de dados não seja o ideal a ser implantado, como por exemplo: segurança (a redundância aumenta a segurança contra perda de dados), dificuldades de controle de acessos aos dados oriundos de diferentes usuários e etc. Desta forma, o objetivo ideal ainda é a criação de diversos bancos de dados na empresa, e admitir um certo nível de redundância controlada [CHU83].

Segundo [CER95], a constante atenção com a flexibilidade e performance para o desenvolvimento de sistemas a integração de dados não somente por setores, mas entre todos os setores da empresa são as principais funções relativas à abordagem de Banco de Dados Relacional. Consiste, fundamentalmente, num agrupamento de todos os dados de interesse de uma empresa, organização e com inter-relação bem estabelecidas, capazes de suprir às exigências de informações de todas as aplicações, eliminando suas dependências em relação à representação física dos dados. Mas existe um problema: a vida dos arquivos de dados, suas interligações, a passagem de ida e volta dos dados para as aplicações, tornaram-se tarefa demasiadamente complexa para estar embutida em cada programa de aplicação, fato agravado pela inadequação para isso das tradicionais linguagens de programação estruturadas. Surgiu, então, e a necessidade de retirar os problemas específicos das aplicações dos problemas de armazenamento e recuperação, que são comuns a todas as aplicações.

Daí então surgem os SGBD, aplicativos voltados para o gerenciamento de Banco de Dados visando tirar os programas de aplicação a responsabilidade de tratar das questões acima mencionadas. Com a abordagem de Banco de Dados Relacional e o SGBD, o desenvolvimento de sistemas passa a se decorrer de maneira mais natural, sem interferir nos sistemas que com ele compartilham dados. As aplicações interagem com a visão lógica do

Banco de Dados Relacional, e a localização física do dado lhe é transparente. Acontece que não existe mais a anterior dependência do programa ou sistema ao arquivo, que agora são tratados como tabelas. Por sua vez, os dados estão integrados, não havendo incoerência de informação, possibilitando maior segurança nos processos operacionais e administrativos [CER95].

2.3.3 SISTEMA DE GERENCIAMENTO DE BANCO DE DADOS

Para [CHU83] um Sistema de Gerenciamento de Banco de Dados (SGBD), é um recurso de software. Na prática, é um conjunto de aplicativos desenvolvido para desempenhar determinadas funções com a finalidade de atingir objetivos específicos envolvendo a formação e utilização de banco de dados.

O SGBD não pode ser considerado como sendo simplesmente um aplicativo oferecido pelos fabricantes de hardware ou fornecedores de software. A sua utilização como software na administração das funções consideradas como recurso das empresas, dentro do contexto de sistemas de informação tem variado em função das características do próprio SGBD.

A existência de linguagens conversacionais existentes como componentes-padrões nos SGBD, permite o desenho de sistemas de informação onde o usuário pode interagir mais intensamente com o aplicativo, na introdução de questionamentos efetuados momentaneamente. Sem dúvida alguma, com SGBD, um sistema de informação é muito mais flexível na consulta de informações.

Por fim, o SGBD teria como finalidade administrar banco de dados contendo todas os dados da empresa, e na sua utilização mais relevante, o SGBD é utilizado como um método de acesso inovador e eficiente no sistema de recuperação de informação (*information retrieval*).

A evolução dos SGBD no mercado tem mostrado que alguns dos SGBD atualmente têm sido utilizados nas versões iniciais como sistema de “*information retrieval*”. No decorrer do tempo, acrescentaram-se outras funções no sentido de torná-lo um sistema genérico de administração de banco de dados [CHU83].

2.3.3.1 ORACLE SERVER

Segundo [RAM99], o Oracle *Server* é um SGBD relacional de um objeto constituído, além desse banco de dados, de uma instância de servidor Oracle. Um Banco de Dados Oracle tem uma estrutura física e lógica. Sendo essas estruturas no servidor são separadas, o armazenamento físico dos dados pode ser gerenciado sem afetar o acesso às estruturas lógicas de armazenamento. Na estrutura física a mesma é determinada pelos arquivos do sistema operacional que o constituem. Cada Banco de Dados Oracle é formado por três tipos de arquivos: um ou mais *datafiles* (arquivos de dados), dois ou mais arquivos de registro *redo*(conjunto de arquivos que protegem os dados alterados do banco de dados e alojados na memória e que não foram gravados nos *datafiles*) e um ou mais arquivos de controle. Esses arquivos fornecem para as informações do ambiente Banco de Dados Oracle um armazenamento físico real.

Já na estrutura lógica do ambiente Banco de Dados Oracle é determinada por um ou mais *tablespaces* (espaços lógicos de armazenamento) e pelos objetos de esquema do banco de dados. Um esquema é uma coleção de objetos os quais por sua vez são as estruturas lógicas que referem diretamente aos dados do banco de dados. Os objetos de esquema incluem estruturas tais como: tabelas, visões, seqüências, procedimentos armazenados, sinônimos, índices, agrupamentos e links de banco de dados. As estruturas de armazenamento lógico, incluindo os *tablespaces*, os segmentos e as extensões, dizem como é usado o espaço físico de um banco de dados. Os objetos de esquemas e os relacionamentos entre eles formam o projeto relacional de um banco de dados.

Ainda, segundo [RAM99], toda vez que se inicia um banco de dados, uma *System Global Area* (área global de sistema) – SGA é alocada e os processos de segundo plano do ambiente Banco de Dados Oracle são inicializados. O SGA é uma área que pertence a memória usada para as informações que são compartilhadas pelos usuários do banco de dados. A combinação dos processos de segundo plano e dos *buffers* (área de armazenamento) de memória é denominada instância Banco de Dados Oracle.

2.3.3.2 TRANSAÇÃO

Para [KOR95], transação é a parte do aplicativo que faz acesso e atualiza vários itens de dados. Cada um desses itens é buscado precisamente uma vez pela transação e é gravado no máximo uma vez se o item de dado for atualizado. A transação não irá violar qualquer restrição de consistência do banco de dados, isto é, quando o banco de dados foi implementado, o mesmo previu que para a transação terminar com sucesso, a mesma deve estar consistente. Mas, durante a execução de uma transação, pode ser necessário temporariamente permitir a inconsistência, porém, mesmo que essa inconsistência seja necessária, poderão ocorrer falhas.

2.3.3.3 ESTADOS DE TRANSAÇÕES

Segundo [DAT91], a transação pode estar no estado ativo (estado inicial), ou no estado parcialmente compromissado (após a última transação ter sido executada), ou no estado falhado (após descoberto que uma execução normal não poderá prosseguir), ou no estado abortado (a transação é desfeita e os dados restaurados ao seu estado anterior ao início da transação), ou ainda no estado compromissado (a transação foi completada "com sucesso"). A operação de *commit* (efetivar), confirma o término de uma transação, ou seja, informa ao aplicativo que um trabalho foi completado com sucesso, estando consistente, e que as atualizações feitas podem ser gravadas ou tornarem-se permanentes. A operação de *rollback* (desfazer), contrariando o *commit*, informa o cancelamento de uma transação, ou seja, informa ao aplicativo que as atualizações executadas falharam, estando inconsistentes, e que todas as mesmas devem ser desfeitas até o estado anterior ao início da transação.

2.3.4 USUÁRIO DE BANCO DE DADOS

Para [CHU83], no desenvolvimento de tecnologias utilizando telecomunicações, vem propiciando a implantação e popularizando a utilização de terminais, junto à própria maturidade e evolução dos sistemas de informação nas empresas, oriunda o surgimento de variados tipos de usuários na busca de informações inseridas num banco de dados.

Pode-se classificá-los nos seguintes tipos mais relevantes:

a) administrador de banco de dados (DBA): é o elemento que possui maior conhecimento e responsabilidade sobre o banco de dados e para tanto, deve possuir o maior nível de capacidade na manipulação de dados, através da Linguagem de Definição de Dados (DDL) e Linguagem de Manipulação de Dados (DML) e, além disso, deve-se preocupar com os aspectos de integridade, segurança e privacidade dos dados do banco de dados e uma gama de aplicativos de serviços e utilitários disponíveis através do SGBD. Além dos aspectos técnicos, o administrador do banco de dados tem a necessidade de conhecer os aspectos importantes das informações;

b) administrador de dados (AD): define o esquema externo da empresa, esse esquema tem que ser consistente e derivável de um único esquema conceitual. Ao DBA e ao AD cabem as seguintes funções : apoio técnico no planejamento, desenvolvimento do modelo de dados e projeto físico do banco de dados, operação do dicionário de dados, implantação de banco de dados (teste e produção), especificações dos procedimentos de segurança e recuperação, desenvolvimento das macro-visões, documentação e manutenção de banco de dados;

c) programador de aplicação: O programador de aplicação, o qual seus conhecimentos e habilidades o qualificam para desenvolver programas utilizando linguagens de programação, em conjunto com as linguagens de definição e manipulação de dados (DDL e DML). Está capacitado a manipular informações contidas no banco de dados;

d) usuário interativo: normalmente trabalha com uma linguagem interativa de consultas, a qual, com base em instruções simples, manipula os dados do banco de dados numa sessão de perguntas e respostas até conseguir (ou não) os resultados desejados;

e) usuário paramétrico: esse usuário não possui tanta liberdade para trabalhar com sessão de perguntas e respostas com o sistema. O sistema mostra um questionário de múltipla escolha (menus), o usuário informa a resposta através de um ou dois comandos e aguarda a resposta. Ao contrário do programador de aplicações, o

usuário paramétrico e interativo são pessoas que não tem conhecimento técnico sobre o funcionamento interno do banco de dados e SGBD. Os mesmos têm acesso para consultar e atualizar dados no banco de dados através de programas desenvolvidos e operacionais. A diferença entre eles está na capacidade de interagir-se com o sistema.

2.3.4.1 LINGUAGEM DE DEFINIÇÃO DE DADOS - DDL

De acordo com [RAM99] as declarações DDL definem, atualizam e liberam objetos quando eles não são mais necessários. Incluem também declarações que permitem que um usuário conceda a outros os privilégios, ou direitos, de acesso ao banco de dados e a objetos específicos dentro do banco de dados.

A DDL é utilizada tanto para definir o conteúdo do banco de dados inteiro como partes do banco de dados de interesse a aplicativos isolados. A definição envolve tanto os diferentes níveis de grupos de dados, como as uniões entre os dados, os quais devem interagir através das diversas ocorrências de valor dos dados.

2.3.4.2 LINGUAGEM DE MANIPULAÇÃO DE DADOS - DML

De acordo com [RAM99], as declarações DML manipulam os dados do banco de dados. Operação de DML é a consulta, inserção, atualização e exclusão de linhas de uma tabela. Também são operações de DML o bloqueio de uma tabela ou do modo e o exame do plano de execução de uma declaração SQL.

DML é utilizada por desenvolvedores de programas para, como a nomenclatura diz, manipular os dados entre o programas de aplicação e o banco de dados. A DML não constitui e nem funciona como uma linguagem independente, o mesmo necessita de uma linguagem na forma de programa residente. Os programas residentes possuem portanto, a parte de procedimento e utilizam a DML para acessar o banco de dados, isto é, para consultar, incluir, excluir e atualizar os dados.

2.3.5 MODELAGEM DE DADOS

Segundo [COR99], modelagem de dados é o processo de desenvolver uma representação de acordo com a visão do usuário referente aos seus dados. É a tarefa mais importante no desenvolvimento das aplicações de um banco de dados eficaz.

2.3.5.1 MODELOS LÓGICOS

Para [KER94] os sistemas de banco de dados no mercado adotaram, geralmente, o modelo hierárquico, ou o modelo de rede, ou o modelo relacional. Dentre esses, o modelo relacional é, com grande margem, o mais difundido.

2.3.5.2 MODELO LÓGICO HIERÁRQUICO

Conforme [KER94], os sistemas de banco de dados no modelo hierárquico foram os primeiros sistemas comercializados. Nesse modelo, o usuário visualiza o banco de dados como uma estrutura de árvore envolvendo registros e relacionamentos. Cada registro pode ter qualquer número de descendentes, mas apenas um ascendente, com exceção do topo, que não tem ascendente. O registro de maior hierarquia guarda referências do conjunto de descendentes que possui.

2.3.5.3 MODELO LÓGICO DE REDE

Para [KER94], a visão que o usuário tem de banco de dados, no modelo de rede é um grafo ou uma malha de relacionamentos, um-para-muitos entre os registros. Um tipo de registro pode estar envolvido em mais de uma ligação, pode ter vários ascendentes e vários descendentes.

2.3.5.4 MODELO LÓGICO RELACIONAL

Para [KOR95], modelo relacional diferencia-se dos modelos hierárquicos e de redes pois esse modelo não usa ponteiros ou ligações. Ao invés disso, o modelo lógico relacional relaciona registros a partir dos valores que os mesmos contêm. Por não utilizar ponteiros, o modelo permite definição de um fundamento matemático formal.

De acordo com [KER94], um banco de dados relacional é visualizado pelos usuários como um conjunto de tabelas. Uma tabela ou relação é composta por linhas denominadas *tuplas* e colunas. A representação por um conjunto de tabelas difere em relação ao modelo de rede, que é um conjunto de registros e relacionamentos através de ligações.

2.3.5.5 MODELO RELACIONAL

Conforme [KOR95], os primeiros sistemas de banco de dados basearam-se nos modelos de redes ou no hierárquico. Esses dois modelos estão ligados mais intimamente à implementação do banco de dados comparando-se ao modelo relacional.

Já para [CHU83], o modelo relacional utiliza-se da teoria matemática relacional como base. Isso significa que os fundamentos da teoria das relações podem ser aplicados no desenvolvimento do modelo de banco de dados.

Para [KER94], modelo relacional é uma maneira de mostrar a organização de um banco de dados como uma coleção de tabelas e associações entre tabelas. Uma relação é uma tabela bidimensional onde cada linha na tabela contém dados que pertencem a alguma coisa ou parte desta. Cada coluna contém dados relativo a um atributo. As linhas das tabelas são também chamadas de *tuplas* e as colunas de *atributos*, conforme demonstrado no quadro 1.

Exemplo de tabela:

Quadro 1: Relação (denominação da tabela): Aluno

	Atributo1 – Nome	Atributo2 – Idade	Atributo3 – Sexo	Atributo4 - Série
Tupla 1	Caroline Scoz	9	F	4
Tupla 2	Felipe Wolff	7	M	2
:	Jamille Martins	5	F	1
Tupla 7	Antônio da Silva	15	M	7

Fonte: [KER94]

2.3.5.6 MODELO ENTIDADE-RELACIONAMENTO

Segundo [KOR95], o modelo entidade-relacionamento (MER) basea-se na visão do mundo real, o qual consiste em um conjunto de objetos denominados entidade e nos relacionamentos entre eles. Esse modelo foi desenvolvido para facilitar a criação de banco de dados permitindo assim a especificação de uma estrutura de uma empresa. Essa estrutura, denominado esquema, representa a estrutura lógica geral do banco de dados.

Para [CHU83], esse modelo têm as seguintes vantagens:

a) instrumento lógico de banco de dados, ao nível do empreendimento como um todo, sem levar em considerações as restrições e características de um sistema de gerenciamento de banco de dados específico;

b) o modelo conceitual atua em um nível de informação com maior estabilidade do que as estruturas lógicas dos usuários, devendo ser pouco afetado nas mudanças das necessidades de informação ou alteração do sistema de gerenciamento do banco de dados da empresa. Dessa forma, alterações realizadas na estrutura lógica física não afeta a estrutura lógica conceitual;

c) diagramação especial do modelo entidade-relacionamento de fácil entendimento, formando assim um instrumento comum de análise para usuários e analistas;

d) fase independente de projeto lógico de banco de dados podendo diagramar modelo entidade-relacionamento com maior facilidade, oriundos de outros modelos citados.

2.3.5.7 MODELO ORIENTADO A OBJETO

Para [SAL95], o modelo orientado a objeto trabalha tudo como sendo um objeto, o qual pode ser manipulado ou inserido em outros objetos manipuláveis. De outro modo, a orientação a objeto resolve alguns problemas do modelo relacional, mais específico ao se tratar com tipos de dados altamente variáveis, como tipo texto. Esse modelo é ideal para

bancos de dados especializados, como sistema de gerencial de documentos, projetado com uma visão voltada ao gerenciamento de documentos baseados em textos.

Segundo [KOR95], o modelo de dados orientado a objeto é uma adaptação da programação orientada a objeto para sistemas de bancos de dados. Em conseqüência um novo tipo de SGBD, o SGBD Orientado a Objeto (SGBDOO) foi desenvolvido. Enquanto o SGBDOO apresenta vantagens significativas sobre o SGBD, a maioria dos dados nos sistemas de hoje estão no formato relacional. As organizações são relutantes em gastar tempo e dinheiro necessários para converter seus dados relacionais para o formato orientado a objeto.

Segundo [COA92], esse modelo é relativamente novo e tende a ser utilizado cada vez mais no desenvolvimento dos sistemas. Para isso é necessário ter domínio da complexidade desse novo paradigma, conforme a seguir:

a) abstração: consiste na seleção de aspectos relevantes e ignorando os irrelevantes;

b) encapsulamento: agrupa aspectos relacionados, minimiza o fluxo entre diferentes partes do trabalho e separa certos requisitos específicos que outras partes da especificação podem usar;

c) associação: agrupar aspectos sob circunstâncias similares;

d) comunicação de mensagens: corresponde aos comandos ou solicitações dos aplicativos;

e) métodos de organização: diferenciado (baseado em experiências), distinção entre objetos como um todo e entre suas partes componentes, formação diferentes classes de objetos;

f) escala: considerar algo muito grande, através de parte ou enfoque total;

g) categorias de comportamento: base na causa imediata, evolução histórica ou similaridade de função.

2.3.6 MAPEAMENTO DO MODELO DE OBJETO PARA RELACIONAL

Segundo [FUR98], existem muitas similaridades entre modelo orientado a objeto e modelo relacional como a estrutura de uma tabela relacional corresponde à estrutura de atributos de uma classe, um índice de uma tabela corresponde a um índice aplicado para todos ou parte da estrutura de atributos de uma classe. Uma visão do modelo de entidades e relacionamentos corresponde aos pacotes e às interfaces de classe que ditam como a classe é acessada e a ocorrência de classes processadas por um usuário ou classe de usuários. Com isso pode-se visualizar um modelo projetado para ser implementado em um ambiente orientado a objeto ser convertido para um ambiente relacional ou vice-versa. Para ser criado um ambiente de banco de dados objeto relacional deve-se ter em mente além do modelo relacional o conceito de orientação a objeto, o modelo de dados deve incluir classe com atributos, operações e restrições de integridade. Uma classe serve como um modelo para instâncias que são criadas compartilhando atributos e operações. O domínio de um atributo é um tipo primitivo de dados, um tipo abstrato de dados ou referência a uma classe, entretanto pode armazenar no máximo um valor. Uma operação é um método que se aplica a cada instância da classe e executa processamento baseado nos valores dos atributos, sendo que o desenvolvedor pode implementar uma operação para anexar a uma classe (encapsulamento).

Restrições de integridade incluem restrições primitivas baseadas no modelo relacional como a especificação de um valor nulo para um atributo, restrição de identificador único para instâncias de classe e restrição de identificador em um ou mais atributos de classe. Um identificador normalmente é implementado como um identificador lógico de um objeto ao invés de seu endereço físico, como ocorre em modelos hierárquicos e redes. O ambiente objeto relacional deve suportar uma linguagem de banco de dados extensiva ao padrão relacional de linguagem estruturada (SQL) para tratar objetos que são definidos e criados usando modelagem orientado a objeto.

Para [COU97], a visualização das abstrações entre os ambientes relacional e orientado a objeto, nos levam a um mesmo caminho durante a modelagem. No quadro 2, pode-se

comparar as equivalências entre os termos utilizados no ambiente relacional e no ambiente objeto relacional.

Quadro 2: Equivalência entre termos relacionais e orientado a objeto

Termos	**Modelo Relacional**	**Modelo Orientado a Objeto**
Instâncias	Elementos individualizados ou Ocorrências	Objetos
Conjunto formado pela agregação de instâncias semelhantes	Entidade	Classe
Características das instâncias dos conjuntos	Atributo	Atributo
Envolvimento entre as instâncias dos conjuntos	Relacionamento	Associação

Fonte: [COU97]

Essa estratégia de incorporar a orientação a objeto juntamente com relacional oferece liberdade de continuar a utilizar aplicações desenvolvidas com técnicas relacionais e integrar-se com novas aplicações utilizando técnicas de orientação a objeto.

2.3.7 UML – LINGUAGEM DE MODELAGEM UNIFICADA

Segundo [BAR98], a *Unified Modeling Language* (UML) é uma tentativa de padronizar a modelagem orientada a objetos de modo geral, com consistência, fácil de se comunicar com outras aplicações, simples de ser atualizado e compreensível.

Existem várias metodologias de modelagem orientada a objetos. Com o agrupamento das melhores idéias de três dessas metodologias, surgiu a UML.

2.3.7.1 OBJETIVOS DA UML

A UML tem como objetivo modelar sistemas utilizando conceitos da orientação a objetos, unir todos os melhores métodos conceituais, criar uma linguagem de modelagem usável tanto pelo homem quanto pela máquina.

Essa modelagem foi criada para ser uma linguagem de modelagem comum a ser usada nas indústrias. Ela está totalmente baseada em conceitos e padrões extensivamente testados provenientes das metodologias existentes anteriormente, e também é muito bem documentada com toda a especificação da semântica da linguagem representada em meta-modelos [BAR98].

2.3.8 FASES DO DESENVOLVIMENTO EM UML

Segundo [BAR98], existem cinco fases no desenvolvimento de sistemas de um aplicativo:

a) análise de requisitos: essa fase busca as necessidades dos usuários do sistema a ser desenvolvido utilizando funções denominadas "*use-cases*". Através do desenvolvimento de "*use-case*", as entidades externas ao sistema (em UML chamados de "atores externos") que interagem e possuem interesse no sistema são modelados entre as funções que eles requerem. Os atores externos e os "*use-cases*" são modelados com relacionamentos que possuem comunicação associativa entre eles ou são desmembrados em hierarquia. Cada "*use-case*" modelado é descrito através de um texto, e este especifica os requerimentos do ator externo que utilizará este "*use-case*". O diagrama de "*use-cases*" mostrará o que os atores externos, ou seja, os usuários do futuro sistema deverão esperar do aplicativo, conhecendo toda sua funcionalidade sem importar como esta será implementada. A análise de requisitos também pode ser desenvolvida baseada em processos de negócios, e não apenas para sistemas de software;

b) análise: a fase de análise está preocupada com as primeiras abstrações (classes e objetos) e mecanismos que estarão presentes no domínio do problema. As classes

são modeladas e ligadas através de relacionamentos com outras classes, e são descritas no Diagrama de Classe. As colaborações entre classes também são mostradas neste diagrama para desenvolver os "*use-cases*" modelados anteriormente. Estas colaborações são criadas através de modelos dinâmicos em UML. Na análise, só serão modeladas classes que pertençam ao domínio principal do problema do software, ou seja, classes técnicas que gerenciem banco de dados, interface, comunicação, concorrência e outros não estarão presentes neste diagrama;

c) *design* (projeto): nessa fase, o resultado da análise é expandido em soluções técnicas. Novas classes serão adicionadas para prover uma infra-estrutura técnica: a interface do usuário e de periféricos, gerenciamento de banco de dados, comunicação com outros sistemas, dentre outros. As classes do domínio do problema modeladas na fase de análise são mescladas nessa nova infra-estrutura técnica tornando possível alterar tanto o domínio do problema quanto a infra-estrutura. O *design* resulta no detalhamento das especificações para a fase de programação do sistema.

d) programação: nessa fase, as classes provenientes do *design* transformam-se em código da linguagem orientada a objetos escolhida. Dependendo da complexidade da linguagem utilizada, essa conversão pode ser uma tarefa fácil ou muito complicada. Nas fases anteriores, os modelos criados são o significado do entendimento e da estrutura do sistema, então, no momento da geração do código onde o analista conclua antecipadamente sobre modificações em seu conteúdo, seus modelos não estarão mais demonstrando o real perfil do sistema. A programação é uma fase separada e distinta onde os modelos criados são convertidos em código.

e) testes: um sistema normalmente é executado em testes de unidade, integração, e aceitação. Os testes de unidade são para classes individuais ou grupos de classes e são geralmente testados pelo programador. Os testes de integração são aplicados já usando as classes e componentes integrados para se confirmar se as classes estão cooperando uma com as outras como especificado nos modelos. Os testes de

aceitação observam o sistema como uma “caixa preta” e verificam se o sistema está funcionando como o especificado nos primeiros diagramas de “*use-cases*”.

2.3.9 COMPOSIÇÃO DA UML

Segundo [BAR98], a UML é composta de:

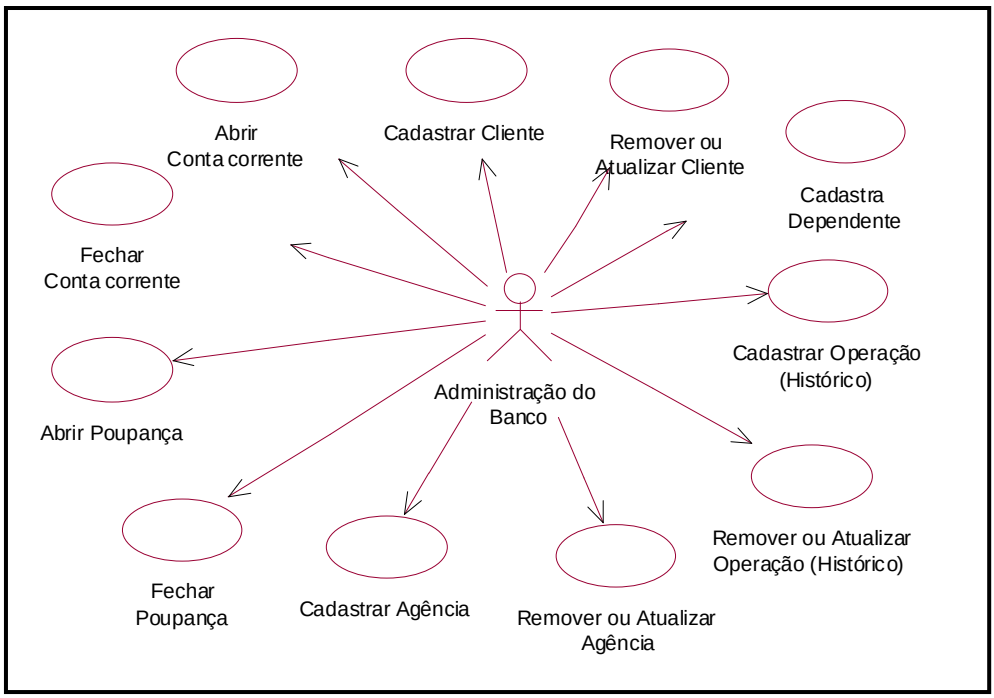
a) visões: mostram diferentes aspectos do sistema modelado. A visão não é um gráfico, mas uma abstração consistindo em uma série de diagramas. Definindo um número de visões, cada uma mostrará aspectos particulares do sistema, dando enfoque a ângulos e níveis de abstrações diferentes e uma figura completa do sistema poderá ser construída. As visões também podem servir de ligação entre a linguagem de modelagem e o método/processo de desenvolvimento escolhido. São cinco as visões: 1) visão “use-case”, a qual descreve a funcionalidade do sistema desempenhada pelos “atores externos” do sistema (usuários), conforme figura 2. 2) visão lógica, descreve a funcionalidade do sistema que será implementado, especificando estrutura estática do sistema como classes, objetos e relacionamentos, e as colaborações dinâmicas no caso de envio de mensagens. 3) visões de componentes, descreve a implementação dos módulos e suas dependências. 4) visões de concorrências, essa visão não é funcional ao sistema pois descreve processos e processadores através dos processos concorrentes (*threads*). 5) Visão organizacional, descreve a organização física do sistema, como conexões entre computadores, periféricos entre outros;

b) modelos de elementos: Os conceitos usados nos diagramas são modelos de elementos que representam definições comuns da orientação a objetos como as classes, objetos, mensagem, relacionamentos entre classes incluindo associações, dependências e heranças.

c) mecanismos gerais: servem para comentários suplementares, informações, ou semântica sobre os elementos que compõem os modelos. Servem também como mecanismos de extensão para adaptar ou estender a UML para um método/processo, organização ou usuário específico.

d) diagramas: são os gráficos que descrevem o conteúdo em uma visão.

Figura 2: Exemplo de diagrama de “use-case”



Fonte: [BAR98]

2.3.10 NOTAÇÃO DA UML

Segundo [FUR98], as principais notações de UML são:

a) pacote: técnica útil para manipular grandes porções de um sistema. É um mecanismo de propósito geral para organizar grandes elementos de modelo em grupos, podendo esse estar inserido em outros pacotes.

b) estereótipo: técnica para criação de uma metaclassificação de elementos no UML, ou seja, incluir novos elementos no modelo, permitindo assim a extensão da capacidade de modelagem da linguagem.

c) nota: utilizado como comentário incluído em um diagrama sem qualquer conteúdo semântico.

As fases de análise de requisitos, análise e *design* (projeto) utilizam em seu desenvolvimento cinco tipos de visões, nove tipos de diagramas e vários modelos de elementos, os quais são utilizados na criação de diagramas e mecanismos gerais que todos em conjunto especificam e exemplificam a definição do sistema.

2.4 BANCOS DE DADOS RELACIONAIS EXISTENTES NO MERCADO

Segundo [TIR98], existem no mercado diversos tipos de bancos de dados relacionais comercializados. Dentre eles alguns serão descritos conforme a seguir:

2.4.1 ADABAS

Esse banco de dados tem como fabricante a Software AG, com sede em Darnstadt, Alemanha. Surgiu em 1971, tendo como característica uma tecnologia que combina funcionalidade e gestão avançada de dados multiplataforma. É desenhado como um sistema aberto e portável entre as plataformas OS/2, Windows NT e Unix. Integridade com ferramentas mais usadas do Windows, como Word e Excel, acesso via ODBC (Open Database Connectivity), permitindo assim fácil recuperação de informações. É compatível com o padrão SQL/ANSI e fornece compatibilidade SQL para DB/2 e Oracle.

2.4.2 ACCESS

O Access tem como fabricante a Microsoft, atualmente encontra-se na versão 2000 para plataformas Windows e Windows NT. Além de fornecer acesso a um banco de dados relacional utiliza-se da linguagem de programação Visual Basic, ferramenta assistente de publicação Web, permitindo criações de páginas na *intranet/internet*.

2.4.3 DATAFLEX

O banco de dados Dataflex foi fabricado pela *software house* americana Data Access, surgindo em 1981. Criado inicialmente como um produto de manipulação de arquivos de dados com uma pequena linguagem de programação, a qual permitia manipular dados de forma rápida e prática. Atualmente o Dataflex constitui-se de uma ferramenta para

desenvolvimento de aplicações completa. O banco de dados é portável nas plataformas Vax/Vms, Dos, Ms-windows, OS/2, Unix-V, Aix e Novell Netware.

2.4.4 PROGRESS

Progress é um banco de dados relacional produzido pela Progress Software Corporation de Bedford, Estados Unidos. Completo ambiente de desenvolvimento de aplicações baseado em componentes oferece ferramentas de desenvolvimento e ambiente de sistemas de classe corporativa através de ambientes heterogêneos e cliente/servidor. O Progress é portável para as plataformas Unix, Windows 95, Windows NT, NetWare NLM, OS/400, Ms-dos, VMS, CTOS e OS/2.

2.4.5 SYBASE

Esse banco fabricado pela Sybase Inc., a qual foi fundada no final de 1984, nos EUA. Tendo como característica básica a arquitetura cliente/servidor, está disponível para mais de 30 plataformas de hardware e software dentre elas destacam-se HP-UX, RS/6000 AIX da IBM, SunOS e Solaris da Sun, VAX/VMS e Alpha /VMS da Digital, NetWare da Novel, Windows da Microsoft.

2.4.6 ORACLE

Segundo [MOR95], a empresa Oracle Corporation teve início no ano de 1969 quando um cientista da empresa IBM, chamado E.F.Codd, revolucionou a área de informações de sistemas com a proposta de uma nova abordagem no gerenciamento de dados.

Dez anos após, Codd introduziu o modelo relacional, essa tecnologia foi lançada na área comercial. Inicialmente a empresa Oracle era composta por quatro desenvolvedores e um contratado. Ao longo do percurso, a empresa Oracle lançou uma série notável de tecnologias inovadoras, conforme abaixo:

Ano	Inovação
1983	Primeiro RDBMS (Sistema de Gerenciamento de Bancos de Dados Relacional) portável;

	Primeiro RDBMS que suporta processadores simétricos;
1984	Primeiro RDBMS baseado em SQL para microcomputadores;
1985	Primeiro RDBMS para ambiente cliente-servidor;
1986	Primeiro RDBMS com capacidade para consulta distribuída;
1987	Primeiro RDBMS para rede de microcomputadores;
1988	Primeiro benchmark TP1 com 100 tps (transações por segundo);
1990	Primeiro servidor de banco de dados para Macintosh;
1991	Primeiro servidor RDBMS paralelo em ambiente “open archive cture”;
	Primeiro benchmark TPC3 acima de 1000 tps (transações por segundo);
	Primeiro RDBMS a ser certificado como 100% compatível com NIST;
1992	Primeiro RDBMS NLM (Netware Loadable Module) certificado pela Novell;
	Primeiro benchmark TPC/A para sistemas abertos acima de 500 tps (transações por segundo);
	Primeiro banco de dados “*cooperative-server*”;
1994	Incorporação e banco de dados distribuído e recurso “*Standby*”;
1995	Incorporação da metodologia orientado a objeto no banco de dados relacional, OLTP (Processamento de Transação on-line);
1998	Incorporação da linguagem universal (Java) no banco de dados e WebDB.

3 QUALIDADE DE SOFTWARE

De acordo com [JUN99], software com qualidade é o mesmo estar em conformidade com os requisitos dos usuários, antecipação e satisfação dos desejos dos clientes, fazer o que está descrito no manual que acompanha o produto. Quando há baixa qualidade no software, o problema não está no software em si, mas na forma em que o mesmo é desenvolvido, ou seja, sem seguir regras, normas ou padrões.

Para [PIS96], o software é visto, atualmente, como uma necessidade e não mais como um incremento de tecnologia ou atividade. Hoje em dia são imensas as atividades que dependem do correto funcionamento de softwares.

A baixa qualidade desse, pode provocar perdas incalculáveis em termos financeiros ou até mesmo, dependendo da aplicação, oferecer riscos à população como um todo. Nesse quadro, fica claro a importância em avaliar a qualidade do software, tanto ao decorrer do produto em desenvolvimento quanto do aplicativo final.

Para que um software venha a competir no mercado o projeto deve ter uma relação custo-benefício adequada e os produtos devem ser de alta qualidade. Além disso, a constante melhoria na qualidade de software é essencial para que o fabricante possa conquistar novos clientes, ou até mesmo manter os clientes atuais. Em estudos recentes, mostram-se que as preocupações dos clientes estão voltados à solução, resultados e funcionalidade dos softwares, o que leva aos fabricantes de softwares, o grande desafio de atingir um novo patamar na qualidade e aumentar a competitividade de seu produto [PIS96].

3.1 HISTÓRICO DA QUALIDADE DE SOFTWARE

Segundo [PIS96], o assunto que refere-se a qualidade de software, com o decorrer dos anos, vem aumentando como é demostrado no quadro 3.

Quadro 3: Histórico da qualidade de software

1950	A qualidade era vista simplesmente como uma verificação de que o produto estava pronto.
1960	Incluída a atividade de testes.
1970	Incluiu-se a verificação do cumprimento de normas definidas para as diversas atividades.
1980	O controle da qualidade de software é visualizada como uma atividade de todo o ciclo de vida do desenvolvimento do produto.
1990	A qualidade de software é considerada uma necessidade. Iniciou-se os primeiros trabalhos de publicação de normas da qualidade
2000	Prevê-se que a maioria dos softwares desenvolvidos, não entrarão no mercado sem que passem por rigorosos testes para assegurar a qualidade dos mesmos.

Fonte: adaptado de [PIS96].

3.2 CONTROLE DA QUALIDADE DE SOFTWARE

Segundo [YOU95], um produto de software contém qualidade de software, desde o início, pela capacidade dos fabricantes que o criaram, da tecnologia utilizada e de uma gerencia que tenha supervisionado o processo.

Já para [ROC95], o controle na qualidade do software é um conjunto planejado e sistemático de testes de todas as situações necessárias para fornecer confiança adequada, da qual o produto está de acordo com os requisitos técnicos estabelecidos.

Conforme [PIS96], hoje em dia, a utilização de sistemas da qualidade tem sido baseada, geralmente, nas normas desenvolvidas pela *International Organization for Standardization* - ISO, denominadas de normas série ISO 9000.

3.3 NORMAS DE GARANTIA DA QUALIDADE

Para [GRI94], as normas da série ISO 9000 a gestão e garantia da qualidade representam uma espécie de acordo de vários países para a normalização da qualidade de procedimentos. Essas normas orientam na procura da melhoria da qualidade de produtos, serviços e relacionamento entre clientes e fornecedores, através de diretrizes para a implantação de sistemas da qualidade nas empresas de diversas atividades.

A sigla ISO - *International Organization for Standardization* (Organização Internacional de Padrões), oriunda-se de um grupo internacional de normas, situado em Genebra, Suíça, fundada em 23 de fevereiro de 1947. Atualmente mais de noventa países fazem parte desse grupo, dentre eles, o Brasil é representado pela Associação Brasileira de Normas Técnicas - ABNT.

3.4 QUALIDADE DE PRODUTOS DE SOFTWARE - ISO/IEC 9126

Segundo [JUN99], a norma denominada ISO/IEC 9126 foi publicada em 1991. Sendo uma das mais antigas na área de qualidade de software já traduzida no Brasil em agosto de 1996 como NBR 13596. Estas normas demonstram o conjunto de características que devem ser verificadas em um software para que o mesmo seja considerado um software de qualidade. conforme descrito no item a seguir.

3.5 DESCRIÇÃO DA NORMA ISO/IEC 9126

A norma de qualidade ISO/IEC 9126 é representado por um desdobramento hierárquico das características de qualidade do produto de software, estando bem definido nos seus dois primeiros níveis (características e subcaracterísticas), deixando o terceiro nível de desdobramento (atributos) a critério do avaliador [JUN99].

As características abrangem todos os aspectos de qualidade de software, ou seja, especifica qualquer requisito de qualidade. As características são seis conforme demonstrado no quadro 4.

3.5.1 CARACTERÍSTICAS E SUBCARACTERÍSTICAS DA QUALIDADE

Quadro 4: Características e subcaracterísticas da norma ISO/IEC 9126

Característica	Subcaracterística	Pergunta chave para a subcaracterística
Funcionalidade (satisfaz as necessidades?)	Adequação	Propõe-se a fazer o que é apropriado?
	Acurácia	Faz o que foi proposto de forma correta?
	Interoperabilidade	Interage com os sistemas especificados?
	Conformidade	Está de acordo com as normas, leis, etc.?
	Segurança de acesso	Evita acesso não autorizado aos dados?
Confiabilidade (é imune a falhas?)	Maturidade	Com que freqüência apresenta falhas?
	Tolerância a falhas	Ocorrendo falhas, como ele reage?
	Recuperabilidade	É capaz de recuperar dados em caso de falha?
Usabilidade (é fácil de usar?)	Intelegibilidade	É fácil de entender o conceito e a aplicação?
	Apreensibilidade	É fácil de aprender a usar?
	Operacionalidade	É fácil de operar e controlar?
Eficiência (é rápido e "enxuto")	Tempo	Qual é o tempo de resposta, a velocidade de execução?
	Recursos	Quanto recurso usa? Durante quanto tempo?
Manutenbilidade (é fácil de modificar)	Analisabilidade	É fácil de encontrar uma falha, quando ocorre?
	Modificabilidade	É fácil modificar e adaptar?
	Estabilidade	Há grande risco quando se faz alterações?
	Testabilidade	É fácil testar quando se faz alterações?
Portabilidade (é fácil de usar em outro ambiente?)	Adaptabilidade	É fácil adaptar a outros ambientes?
	Capc.para ser instalado	É fácil instalar em outros ambientes?
	Conformidade	Está de acordo com padrões de portabilidade?
	Capac. para substituir	É fácil usar para substituir outro?

Fonte: Adaptado de [JUN99]

3.5.2 MÉTRICAS DE SOFTWARE

Existe uma área de estudos à parte dentro da qualidade de software denominada de métricas de software. A pretensão é definir, de forma precisa, como medir numericamente uma determinada característica. Essas métricas estão definidas em externas e internas [JUN99].

3.5.2.1 MÉTRICAS EXTERNAS

É uma escala quantitativa e o método que pode ser usado para medir uma característica ou subcaracterística do software independente do comportamento do sistema que contém o software [JUN99].

Essa parte pode ser usada pelos desenvolvedores, avaliadores, compradores e pelos usuários. Cada métrica pode ser aplicada para medição de uma característica da qualidade, uma subcaracterística ou um atributo externo de um produto de software. Essas métricas externas, resumem-se na especificação de requerimentos da qualidade, na avaliação de qualidade de produtos no teste final, e no teste de aceitação. Essa métrica pode ser utilizada como referência para desenvolvimento de novas métricas, e para pesquisas e estudos em geral.

3.5.2.2 MÉTRICAS INTERNAS

É uma escala quantitativa e o método que pode ser utilizado para medir diretamente um atributo ou uma característica do software. Essa parte é especialmente útil para desenvolvedores e alguns avaliadores que podem obter materiais internos tais como especificações e código fonte.

Essa parte proporciona uma coleção de métricas internas e alguns guias para seu uso. Cada métrica pode ser aplicável para medir um atributo interno do produto de software. Também proporciona características internas e modelo da qualidade que mostram a relação entre eles como um guia. Essa métrica é útil como definição dos objetos do projeto e revisão de produtos intermediários, pode ser usada como referência para o desenvolvimento de métricas novas, e para pesquisas gerais e estudos.

A documentação que padroniza as definições de características da qualidade e métricas que são recomendadas para serem usadas na avaliação e especificação dos requisitos da qualidade poderão ser encontradas na ISO/IEC 9126 / NBR 13596, disponíveis através do site na *internet* da Associação Brasileira de Normas Técnicas no endereço eletrônico: www.abnt.com.br para maiores detalhes.

4 TÉCNICAS E FERRAMENTAS UTILIZADAS

As ferramentas utilizadas para o desenvolvimento das aplicações no ambiente Oracle relacional versão 7 e Oracle objeto relacional versão 8 são baseadas em arquitetura cliente-servidor e numa programação orientada a eventos. Deve-se lembrar que parte do processamento se realiza em uma máquina (*client*) e parte se realiza no servidor onde está localizado o banco de dados (*server*). O entendimento deste conceito permite um balanceamento de recursos de processamento obtendo assim uma melhor utilização dos dois ambientes.

4.1 MÉTODOS NO DESENVOLVIMENTO DE SISTEMAS

Segundo [CHU83], a utilização de bons métodos no desenvolvimento estruturado de sistemas se faz necessário para garantir aos usuários sistemas cada vez mais confiáveis, disponíveis em menor tempo e fornecendo informações exatas para que os usuários possam utilizar o aplicativo com maior precisão quando necessário e no momento que quiserem. Uma metodologia deve, portanto, conduzir um caminho nas quais as necessidades deverão ser supridas, como também as operações rotineiras. Ao mesmo tempo, deve ser totalmente flexível e abrangente o bastante para garantir a inclusão de novas técnicas que surgem no mercado, como também se adequar às novas necessidades que aparecem no mercado e que as empresas deverão enfrentar.

Para [FUR98], as empresas de informática vem oferecendo soluções que buscam diminuir dificuldades, simulando modelos da realidade de forma mais simples através de ambientes gráficos e interfaces ricas. O primeiro método de análise orientado a objeto (AOO) empregava o que é atualmente conhecido como análise entidade relacionamento estendido como a base do enfoque. Incorporava tanto herança como agregação. Atualmente a AOO aprimora análise e projeto estruturado, análise essencial e engenharia da informação. Métodos orientados a objeto procuram redefinir e estender métodos existentes, procurando tornarem-se metodologias híbridas (comum entre todos os métodos).

O desenvolvimento de sistemas é uma arte que leva em conta o relacionamento entre seres humanos e computadores. A metodologia de análise estruturada a ser apresentada, numa visão geral, tratam de todo o ciclo de vida de um sistema.

4.2 ANALISE ESTRUTURADA DE SISTEMAS

Segundo [CHU83], o denominado projeto preliminar atualmente passa a ser denominado Desenho Estruturado, ou Análise Estruturada (também chamado Desenho Composto ou Projeto *top-down*).

4.2.1 CICLO DE VIDA DE UM PROJETO ESTRUTURADO DE SISTEMA

Conforme [DAT91], as fases utilizam-se da codificação estruturada, implementação *top-down*, subteste entre outras. Interessa principalmente a fase analítica, mas a migração para esse novo ciclo de vida é relevante porque implica um uso diferente do Documento Final, também conhecido como Documento de Especificação Funcional.

Sendo que a Análise Estruturada produzirá um novo tipo de Documento Final para alterar a Especificação Funcional clássica, deve-se considerar como o produto de análise deverá ser utilizado. A seguir, um exemplo de uma descrição mais detalhada da transformação denominada Análise Estruturada.

A Análise Estruturada subdividi-se em cinco partes:

a) **Pesquisa:** Necessidades do usuário e documento de viabilidade;

b) **Análise:** Requisitos físicos, especificação de funções ou ações, planejamento e orçamento;

c) **Projeto Estruturado:** Projeto empacotado e plano de testes;

d) **Estudo de Computador:** Licitação de máquina e configuração de dados;

e) **Implementação *Top-Donw*:** Resultado de testes e implementação do sistema.

4.2.2 FERRAMENTAS DE UMA ANÁLISE ESTRUTURADA

4.2.2.1 DICIONÁRIO DE DADOS

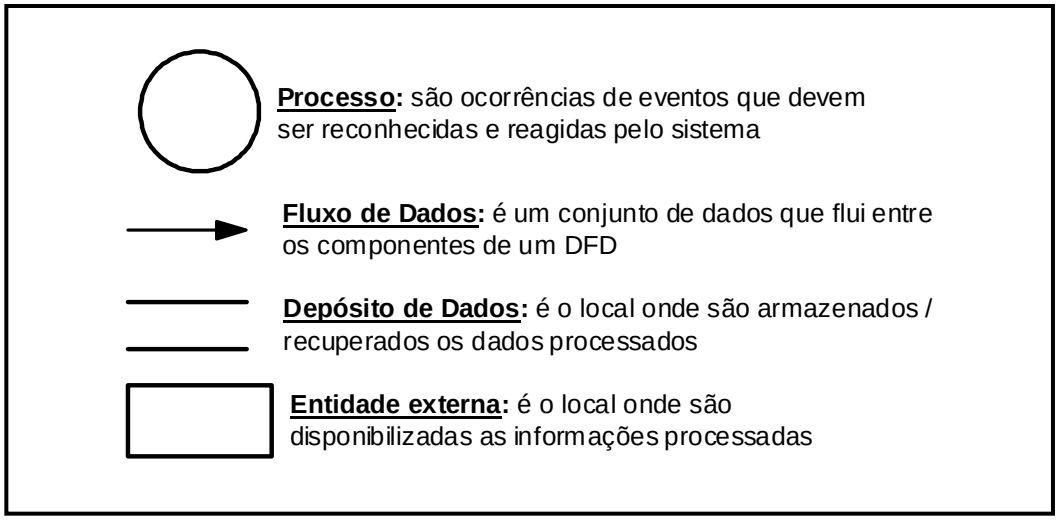
É considerada a ferramenta mais importante na utilização da Análise Estruturada. É importante que o dicionário retrate as entidades, seus relacionamentos e atributos, entre outros elementos, conseguindo dessa forma incorporar as bases de um Gerenciador de Banco de Dados Relacional.

O dicionário de dados deve também consistir de dados e associá-los entre várias tabelas, conter os conceitos de chave primária (simples ou composta), chave unívoca, chave estrangeira, integridade de entidade e integridade referencial, domínios de colunas (incluindo validação desses domínios), documentação de todos os componentes de uma aplicação, e servir de ferramenta principal na Administração de Dados e Administração de Banco de Dados. É relevante que o dicionário de dados seja ativo e dinâmico para interagir com o sistema, como o *CASE*Dictionary* da empresa Oracle.

4.3 DIAGRAMA DE CONTEXTO

Segundo [SHI93], o diagrama de contexto define com o que ou com quem o sistema faz interface e qual o conteúdo daquela interface, ou seja, o diagrama de contexto define de forma quantitativa o limite sistema-ambiente. Existem atividades recomendadas para desenvolver um diagrama de contexto as quais são: criar e modelar a identificação o qual referencia ao sistema, listar e modelar os intervenientes o qual define as funções que fazem interface com o sistema, fluxo de dados que são caminhos vetorizados para os dados, dicionário de dados o qual especifica todas as informações do modelo e glossário que são expressões ou palavras que não são candidatas ao dicionário de dados.

Figura 3: Elementos gráficos do D.F.D., notação Yourdon.

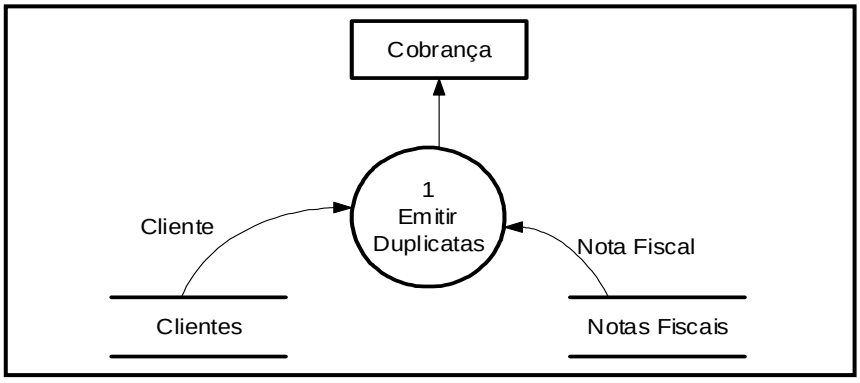


Fonte: adaptado de [SHI93]

4.4 DIAGRAMA DE FLUXO DE DADOS – D.F.D.

Segundo [SHI93], o diagrama de fluxo de dados (DFD) é uma ferramenta para modelagem da perspectiva de atividade. O DFD é verticalmente nivelado, pois cada atividade se decompõe em atividades de nível mais baixo até que não haja mais decomposição. Os componentes gráficos do DFD dizem o que o processo é, sem mostrar como ele funciona. Especificações do processo são requeridas somente para funções primitivas, os outros processos são especificados pelos processos de níveis mais baixo. Esse método minimiza a duplicidade, o que facilita na manutenção. Elementos gráficos da notação Yourdon estão demonstrados na figura 3. Um exemplo de diagrama de fluxo de dados está demonstrado na figura 4.

Figura 4: Exemplo de D.F.D.



4.5 MODELO ENTIDADE X RELACIONAMENTO – M.E.R.

Esse modelo foi proposto por [CHE77], representado por um gráfico denominado diagrama entidade x relacionamento. É um diagrama que detalha as associações existentes entre as entidades de dados e utiliza componentes semânticos próprios.

Esse modelo têm os seguintes aspectos: cada entidade é representada por um retângulo na posição horizontal, cada tipo de relacionamento é representado por um losango, pode haver um tipo de relacionamento entre mais do que duas entidades, os relacionamentos podem conter atributos, pode haver mais de um tipo de relacionamento entre duas entidades.

No MER é necessário considerar cada entidade sob o enfoque de dados, não levando em conta os procedimentos ou as rotinas. Os relacionamentos se fazem entre os dados de uma entidade em relação aos dados das demais entidades que formam o modelo.

Tem como princípio básico a teoria de conjuntos. Cada entidade conceitual deve ser vista como sendo um conjunto que pode ou não ter relacionamento (interseção) com outro conjunto.

4.5.1 TERMINOLOGIA BÁSICA DO MER

Segundo [SHI93], fazem parte da terminologia básica do MER os seguintes elementos:

a) entidades: são os componentes físicos e abstratos utilizados no sistema, onde são armazenados dados;

b) atributos: representam as propriedades de uma entidade;

c) ocorrência: conjunto de atributos de uma determinada entidade;

d) relacionamento: é uma correspondência entre duas ou mais entidades;

e) identificador ou atributo determinante: um atributo ou uma coleção de atributos que determina de modo único uma ocorrência de entidade;

f) grau de relacionamento: o número de entidades que participam da associação;

g) classe de relacionamento ou cardinalidade: quantas ocorrências de cada entidade estão envolvidas no relacionamento. Pode ser:

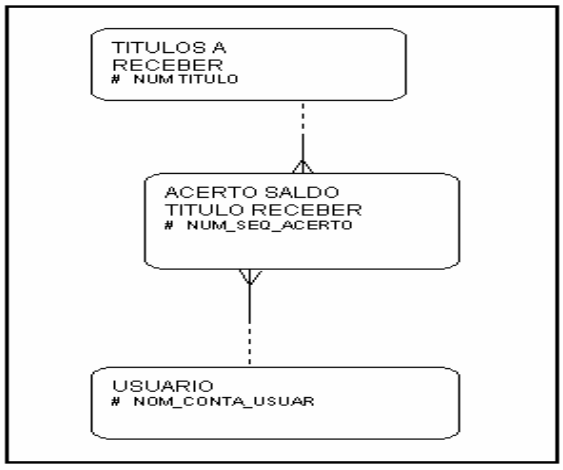
- 1:1 (um para um);

- 1:n (um para muitos);

- m:n (muitos para muitos).

No exemplo da figura 5 podemos verificar que uma ocorrência na entidade “titulo a receber” pode estar relacionada com uma ou mais ocorrências na entidade “acerto saldo título receber”, o que também pode acontecer a mesma situação com a entidade “usuário” relacionando-se com a entidade “acerto saldo título receber”.

Figura 5: Exemplo de M.E.R.



4.6 DIAGRAMA ENTIDADE RELACIONAMENTO – D.E.R.

Segundo [SHI93], o DER é a ferramenta que modela a perspectiva de memória do sistema. Demonstra qual a informação o sistema necessita lembrar para responder a chamadas externas.

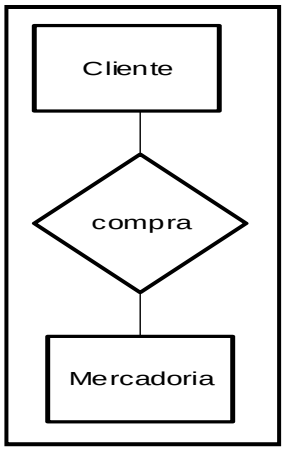
É o diagrama da entidade, as quais devem ter as seguintes características: a) somente classes de entidades possuem atributos, b) toda classe de entidades possui pelo menos um atributo ou conjunto de atributos que sirva como identificador, c) em toda classe de entidades

não haverá atributos que não sejam exclusivos daquela classe. Uma exceção é feita para os atributos que servirem como elementos de ligação (atributos relacionantes ou referenciais) como outras classes de entidades, os quais poderão ser comuns.

No contexto de um sistema, nenhuma classe de entidades a ele pertencente existe isoladamente. Deverá sempre estar ligada a pelo menos uma outra classe de entidade, através de um relacionamento, o qual é materializado através de atributos comuns, como descrito na premissa anterior.

Entre cada duas classes de entidades deverá ser expresso um único relacionamento. Não ocorre auto-relacionamento (relacionamento de uma classe de entidades com ela mesma).

Figura 6: Exemplo de D.E.R.



4.6.1 NORMALIZAÇÃO

Segundo [NET88], normalização é um processo formal passo a passo que examina campos (atributos) de uma tabela (entidade), com o objetivo de evitar anomalias nas inclusões, exclusões e alterações de linhas (tuplas).

4.6.1.1 ROTEIRO DE NORMALIZAÇÃO

Relação não-normalizada (estrutura de dados com grupos repetidos) segundo [CHU83].

a) **1FN** (primeira forma normal): Desmembrar a relação em uma ou mais relações sem grupos repetidos. Definir um ou mais domínios (elementos de dados) como a chave primária: a menor chave que identifique exclusivamente cada *tupla* (ocorrência da estrutura de dados). Desmembramento de grupos de dados em itens elementares (itens atômicos). Relação normalizada na primeira forma normal (sem grupos repetidos, isto é, domínios identificados por chave única e dados atômicos, domínio multivalorado, eliminar repetição). Exemplo: entidade Notas_Fiscais, inclui-se a entidade Itens_Nota_Fiscal para evitar-se ítens repetitivos.

b) **2FN** (segunda forma normal): Para relações cujas chaves tenham mais de um domínio, verificar se cada domínio que não é chave é dependente funcionalmente da chave como um todo e não apenas parcialmente dependente de parte dela. Desmembrar a relação (se necessário) para conseguir isso. Todos os domínios que não são chaves são completamente dependentes funcionalmente da chave primária. Exemplo: entidade Itens_Notas_Fiscais, inclui-se a entidade Produtos.

c) **3FN** (terceira forma normal): Verificar se todos os domínios que não são chaves são independentes entre si. Remover domínios redundantes ou desmembrar a relação (se necessário) para conseguir isso. Todos os domínios que não são chaves completamente dependentes funcionalmente da chave primária e independentes entre si. Exemplo: entidade Nota_Fiscal, inclui-se as entidades Clientes, Vendedores, entre outras.

d) **4FN** (quarta forma normal): Verificar o conjunto de valores dependentes multivalorados. Consegue-se através de um cálculo aritmético, ou através de procedimentos manuais.

e) **BCNF (Boyce Codd Normal Form):** Verificar chaves com elementos sobrepostos. Analisa e organiza as tabelas para que sua estrutura seja simples, relacional e estável.

4.7 ANÁLISE ORIENTADA A OBJETOS

Segundo [BAR94], a análise orientada a objetos (AOO) surgiu para implementar sistemas cuja definição seja especial ou não-convencional, onde incluem-se aplicações de reconhecimentos de padrões, aplicações gráficas, tratamento de imagens entre outros. A AOO veio para ocupar o espaço que os outros modelos convencionais do tipo hierárquico, rede e relacional não conseguem resolver com plenamente esses tipos de sistemas.

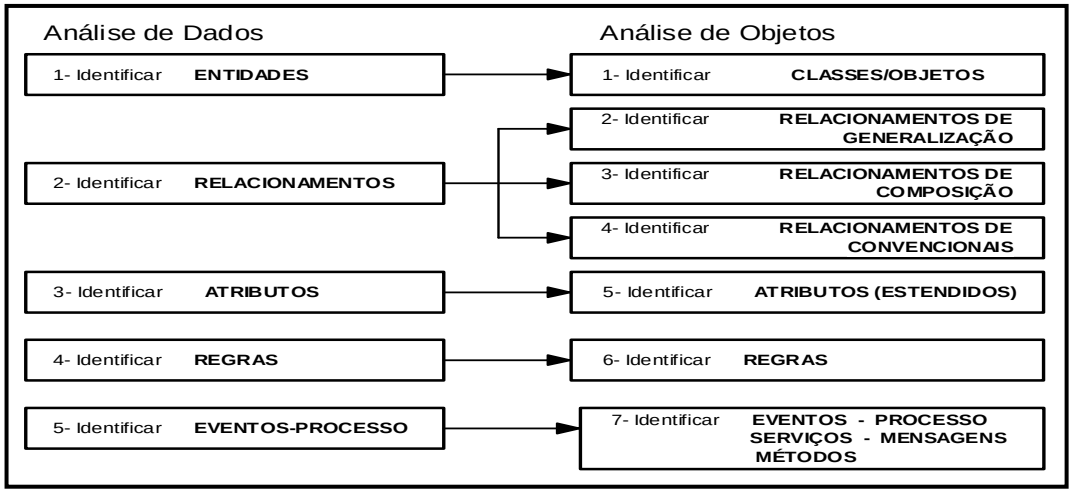
Para [COL96], a análise orientada a objeto deve desenvolver um modelo de objetos para o domínio do problema, determinar a interface do sistema, desenvolver um modelo de interfaces e verificar os modelos de análise.

4.7.1 METODOLOGIA PARA ANÁLISE ORIENTADA A OBJETOS

Métodos de modelagem/análise orientados a objetos começaram a surgir em meados de 1970 e inicio de 1980, e a grande maioria basearam-se em experiências adquiridas com a engenharia da informação e o uso de conceito de caso [COA92].

As aplicações que utilizam AOO, convencionam conceitos estendidos de classe/subclasse (maior refinamento de entidades), eventos, métodos encapsulados e diagrama de serviços (uma nova visão de análise funcional) [BAR94].

Figura 7: Comparativo entre análise de dados e análise de objetos



Fonte: [BAR94].

Os passos básicos na implementação de sistemas utilizando AOO, conforme figura 7, resumem-se em:

a) identificação das classes e seus objetos: classes são coleções de objetos, as quais têm as mesmas características de atributos e de comportamento;

b) identificação de relacionamento de generalização: definem-se as classes e subclasses que possuem comportamento comuns;

c) identificação de relacionamento de composição: também conhecidas como “todo-parte” ou agregação, são identificados como estruturas hierárquicas do tipo “contém-os”, “cabeçalho itens”, entre outros;

d) identificação de relacionamentos convencionais: resume-se em relacionamentos tradicionais (associações);

e) identificação de regras: comportamento semântico das classes e seus objetos, determinadas pelas regras de criação, extinção, modificação e manipulação das ocorrências de objetos;

f) identificação de atributos: composição das classes;

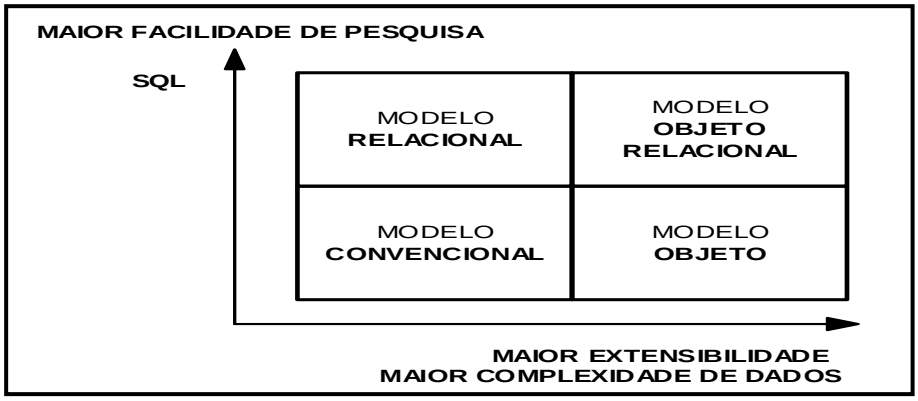
g) identificação de serviços inerentes dos objetos e atributos: serviços que cada classe tem a oferecer, manipulação dos objetos de uma classe, esses métodos estarão “encapsulados” nas classes e serão ativados através de mensagens.

4.7.1.1 MODELO OBJETO RELACIONAL

Segundo [BAR94], o modelo objeto relacional é uma junção de conceitos de objetos com o poder da linguagem SQL, e é uma forma de atribuir aos tradicionais SGBD relacionais o suporte ao conceito de tipos estendidos de dados, regras, classes, entre outros. Isso significa poder implementar sistemas denominados complexos como tratamento de imagens, reconhecimentos de padrões, entre outros utilizando banco de dados relacionais. Esses produtos com modelo objeto relacional estão disputando diretamente com produtos da linha objeto “puro” com mercado ainda em crescimento. Percebe-se então que empresas que

desenvolvem produtos relacionais tradicionais estão voltadas aos conceitos de objetos. As relações (ou os domínios) serão transformadas em classes, as linhas (tuplas) serão os objetos e os atributos poderão ser multi-valorados. Na figura 8 demonstra a atual situação dos modelos relacionais, objetos e objeto relacional em termos de complexidade e facilidades aos acessos de dados.

Figura 8: Posicionamento relativo dos modelos convencionais e estendidos.



Fonte: [BAR94].

4.8 FERRAMENTAS DE ANÁLISE ORIENTADO A OBJETO SEGUNDO UML

4.8.1 DICIONÁRIO DE DADOS

Segundo [COL96], o dicionário de dados é uma ferramenta vital devido a sua função de único local onde as definições podem ser encontradas. A definição dos itens encontrada no dicionário demonstra a clareza do pensamento, além de desempenhar um papel de verificação dos modelos quanto à consistência e o entendimento para as pessoas que não estão familiarizadas com o desenvolvimento do sistema.

4.8.2 DIAGRAMA DE CLASSE

Segundo [FUR98], esse diagrama trata-se de uma estrutura lógica estática em uma superfície de duas dimensões mostrando uma coleção de elementos declarativos de modelo, como classes, tipos com seus respectivos conteúdos e relações. Existem quatro tipos principais de relacionamento de diagrama de classes: a) Generalização/especificação: indica o relacionamento entre um elemento mais geral (superclasse) e um elemento mais específico

(subclasse). b) agregação: utilizado para demonstrar relacionamento todo/parte. c) associação: demonstrar relacionamento entre classes não correlatas, como exemplo: cliente pode comprar várias mercadorias. d) dependência: relacionamento entre um elemento independente e um dependente.

4.8.3 DIAGRAMA DE CASO DE USO

Segundo [FUR98], diagramas de caso de uso fornecem um modo de demonstrar a visão externa do sistema analisado e suas interações com o mundo exterior, mostrando uma visão de alto nível de funcionalidade intencional mediante o recebimento de um tipo de requisição do usuário. Esse diagrama resume-se em um gráfico de atores, conjunto de casos incluído por limite de domínio, comunicação, participação e associações entre atores.

4.8.4 DIAGRAMA DE INTERAÇÃO

Segundo [FUR98], diagrama de interação é um termo genérico que se aplica a vários tipos de diagrama que enfatizam interações de objetos. Uma interação é uma especificação comportamental que inclui trocas de mensagens entre objetos dentro de um contexto para realizar um propósito específico. Em UML, diagramas de interação são demonstrados de duas formas: a) diagrama de seqüência: expõe o aspecto do modelo que enfatiza o comportamento dos objetos em um sistema, e b) diagrama de colaboração: o qual demonstra o contexto completo de uma interação.

4.8.5 DIAGRAMA DE ESTADO

De acordo com [FUR98], esse diagrama é utilizado quando a mudança de um estado em um objeto (início ou termíno de uma determinada tarefa) a ordem em que as operações/atividades são executadas é importante, portanto, nem todos os métodos orientados a objeto utilizam os diagramas de estado, mas é uma técnica expressiva dentro da UML.

4.8.6 DIAGRAMA DE IMPLEMENTAÇÃO

Para [FUR98], a arquitetura física demonstra a composição detalhada do hardware e do software, a qual pode ser representada pelo diagrama de implementação dentre os quais

demonstram dispositivos de hardware, localização física dos componentes, dependências entre arquivos e inclusive arquivos que devem ser recompilados (codificar para código de máquina) quando algum arquivo sofre manutenção. Existem duas formas do diagrama no caso de códigos fontes (arquivos) em UML: a) diagrama de componente: o qual demonstra a estrutura do próprio fonte e b) diagrama de implantação: o qual demonstra a estrutura do sistema de *run-time* (tempo de execução).

4.9 LINGUAGEM DE QUARTA GERAÇÃO - SQL

Segundo [CER95], a padronização de uma linguagem é muito importante, desenvolvedores de computadores e da lógica computacional estão sempre na busca de padrões, sem muito êxito. A necessidade de padronização é grande pois os bancos de dados são os principais repositórios de informação e um acesso padrão é de grande importância para interação dentro de uma organização ou entre várias organizações. É ainda mais importante no ambiente de microinformática, onde o desenvolvimento de bases de dados tem sido aleatório e bancos de dados de diferentes fabricantes podem ser utilizados numa mesma empresa.

Uma linguagem padrão em potencial surgiu para os bancos de dados relacionais denominada Linguagem de Consulta Estruturada (*Strutured Query Language*), conhecida como SQL, que foi desenvolvida para o modelo relacional de E.F. Codd na década de 1970. Essa linguagem suporta transações de processamento de consultas, segurança de administração de dados, integridade e recuperação em caso de perda dos dados. Pelo simples fato de ter sido adotada como linguagem padrão para bancos de dados, a SQL destaca-se no processamento distribuído e na interligação entre computadores. A SQL basea-se nos princípios da álgebra e cálculo relacional.

4.10 FERRAMENTAS CASE ORACLE

De acordo com [NET88], com o crescente desenvolvimento de sistemas informatizados, vêm surgindo cada vez mais ferramentas do tipo *Computer-Aided Software Engineering* (Engenharia de software auxiliada por computador) - CASE, voltadas a dar apoio

e possibilitar o desenvolvimento gráfico, a integração, a interação e a documentação dos sistemas desenvolvidos.

Segundo [CER95], o ambiente Oracle versão 7 possui diversas ferramentas CASE, dentre elas citam-se as seguintes:

4.10.1 CASE*METHOD

O *CASE*Method* é ferramenta para desenho, desenvolvimento e documentação no auxilio do gerenciamento das aplicações do ambiente Oracle, ou qualquer outro banco de dados relacional. Ela incorpora o conceito de arquitetura aberta adotada pela empresa Oracle. Essa ferramenta cria automaticamente uma documentação técnica para as aplicações de qualquer banco de dados relacional através de uma pesquisa ao dicionário de dados. Está direcionado para o ambiente relacional e orientado à objeto, ou seja, a visão externa, a visão conceitual e a visão física. Tem manutenção dinâmica e ativa dos dados, o qual fornece total integração do ambiente. Compõe-se dos seguintes produtos: a) *CASE*Dictionary*; b) *CASE*Designer*; c) *CASE*Generator*; d) *CASE*Exchange*.

4.10.1.1 CASE*DICTIONARY

O *CASE*Dictionary* é parte integrante do Ambiente de Desenvolvimento Cooperativo (CDE - *Cooperative Development Environment*). Foi desenvolvido para auxilio a desenvolvedores de sistemas em suas tarefas de desenvolvimento de aplicativos e suportar os vários novos pacotes do ambiente Oracle surgidos com a versão 7, para cliente/servidor, o qual inclui integridade de dados, segurança estendida, comandos PL/SQL, acesso distribuído, inclui um várias opções para relatórios.

Os diversos tipos de dados (char, varchar2, number, long, date, entre outros) adotados pela ANSI SQL3 fazem parte integrante da definição de dados do *CASE*Dictionary*, e também na Integridade Declarativa de Dados como: a) Não-nulo; b) Padrão de Omissão (Default); c) Unívoco; d) Chave Primária; e) Chave Estrangeira; f) Condição de Verificação; g) Cancelamento de Atualização em cascata; h) Gravar exceções em tabelas especificada; e i) Habilitar/Desabilitar objetos.

4.10.1.2 CASE*DESIGNER

O *CASE*Designer* é uma ferramenta gráfica para auxílio no desenvolvimento de diagramas que atualizam o CASE*Dictionary. Os principais diagramas que podem ser criados são:

a) DER - Diagrama Entidade/Relacionamento;

b) DHF - Diagrama Hierárquico de Funções;

c) DFD - Diagrama de Fluxo de Dados;

d) DM - Diagramas Matriciais Bi-direcionais;

e) DE - Diagrama de Evento.

É portável numa grande faixa de ambientes gráficos ente eles Windows NT, Windows95, Linux entre outros. Totalmente integrado com o *CASE*Dictionary* em equipamento local ou em rede. O *CASE*Designer* é um produto de auxílio para a análise global de informação da empresa, além disso, melhora em grande escala o desenvolvimento de aplicativos.

4.10.1.3 CASE*GENERATOR

O *CASE*Generator* é um gerador de aplicativos que se baseia em informações existentes no *CASE*Dictionary* para gerar programas para os seguintes produtos: Oracle Forms, Oracle Report e SQL*Plus.

Com o *CASE*Generator*, produz-se rapidamente aplicativos baseados no repositório do *CASE*Dictionary*, reduzindo em muito o tempo de desenvolvimento de sistemas, com toda a documentação necessária, garantindo qualidade e melhor controle. Engenharia reversa também é possível de ser realizada analisando-se as informações dos objetos existentes dentro do SGBD Oracle, partindo-se do físico e chegando-se ao conceitual.

4.10.1.4 CASE*EXCHANGE

O *CASE*Excahange* é uma ligação entre o *CASE*Method* e outros CASEs existente no mercado, permitindo acesso e migração. Assim, torna-se viável a exportação e importação de desenvolvimentos com a utilização de outros CASEs para dentro do *CASE*Dictionary*. Os tipos de elementos suportados são: entidades e atributos, relacionamentos, identificadores unívocos e domínios (discretos, contínuos e dimensionais).

4.10.2 ORACLE FORMS

Segundo [CER95], o Oracle Forms é uma ferramenta multimídia de quarta geração a qual permite desenvolver rapidamente e facilmente aplicações em telas, com acesso direto ao dicionário de dados para a geração automática dessas telas e utilização do banco de dados. É uma ferramenta de grande produtividade, com facilidades no aprendizado, tanto por parte do desenvolvedor como pelo usuário da aplicação. Totalmente orientado por meio de janelas e ícones, o Oracle Forms cria aplicações bastante complexas sem a necessidade de desenvolvimento dessas aplicações. As aplicações podem ser criadas ou processadas em qualquer tipo de computador e, além de multiusuárias, permitem consultas, atualizações, inserções ou exclusões de registro à base de dados. Essa ferramenta facilita a prototipagem de aplicações. Vantagens do Oracle Forms: objetivo de interface gráfica para usuários, editor de textos integrado, acesso sem limite a dados, arquivos com formato de imagem e gráficos, capacidade de gerar aplicativos padrões, capacidade para gerar consultas padrões, protótipos de telas, capacidade procedural, suporte de idiomas pátrios, portabilidade de interface a usuário e integração total com outros produtos.

4.10.3 ORACLE REPORTS

Gerador de relatórios utilizando gráficos e textos, formatando os resultados das pesquisas para um relatório personalizado pelo usuário utilizando apenas alguns comandos. Pode ser especificado paginação, cabeçalhos, rodapés, títulos, tamanhos e cores de campos, formatos e outras facilidades, inclusive formulários pré-impressos ou intercalação com textos predefinidos, cartas, circulares, e outras facilidades a mais. Além de tudo isso, ainda pode-se incluir lógica para controlar a execução do relatório. Vantagens do Oracle Report: poderoso

controle de dados, capacidade extensiva de agregação, acesso a dados sem limite, formato de arquivo imagem ou gráfico; pintor gráfico comum, desenhador padrão de relatórios, capacidade procedural, portabilidade de interface a usuário, suporte a idiomas pátrios, integração a produtos, métodos de armazenamento de relatórios e segurança de relatório [CER95.

4.10.4 LINGUAGEM DE PROGRAMAÇÃO ESTRUTURADA PL/SQL

A linguagem PL/SQL foi desenvolvida pela empresa Oracle para executar comandos procedurais, funciona integralmente com a SQL e o servidor Oracle. É de total integração com o banco de dados, mantendo integridade e segurança incontestável. É uma linguagem estruturada de bloco robusta, com capacidade procedural e depurador de erros. Características do PL/SQL: suporte à SQL, programação estruturada em blocos, controle de enlaces, fluxo de controles de comandos, poderoso controlador de erros, integrado com o servidor Oracle, integrado com o Oracle Forms, integrado com o Oracle Reports e integrado com o SQL*Plus e SQL*DBA.

De acordo com [RAM99], a linguagem PL/SQL é uma extensão da linguagem SQL, é um dialeto da SQL especializado no ambiente Oracle. Através dela pode-se criar objetos de esquema como:

a) *triggers*: é uma rotina PL/SQL armazenada no próprio banco de dados e que é executada imediatamente antes ou após comandos de inserção, atualização ou exclusão, conforme programado pelo desenvolvedor.

b) *stored procedures / function*: é um procedimento armazenado no próprio banco de dados. A diferença entre *procedures*(procedimentos) e *functions*(funções) é que as *functions* retornam somente um valor e as *procedures* podem retornar vários ou nenhum valor.

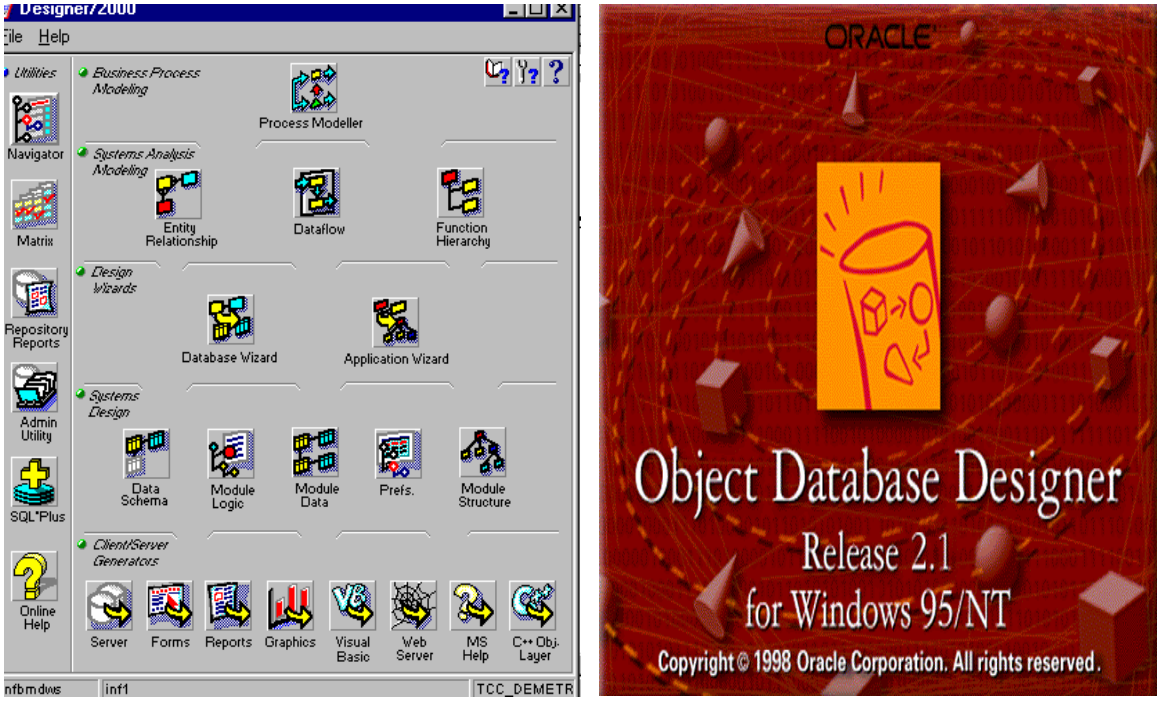
c) *package*: uma *package*(pacote) é um arquivo que agrupa funções, procedimentos e variáveis em um mesmo lugar no banco de dados.

5 DESENVOLVIMENTO DO TRABALHO

5.1 ESPECIFICAÇÃO

No desenvolvimento da aplicação, foram utilizadas as ferramentas de propriedade da empresa Oracle que compõem o Case*Designer 2000, definido nesse trabalho no capítulo 4.11 ferramentas Oracle para criarem as tabelas através do MER e do DFD, foi utilizado também a linguagem de programação PL/SQL do ambiente.

Figura 9: Ferramenta CASE Oracle Designer 2000



Fonte: Produtos Oracle designer 2000 e *Object Database Designer*

5.2 APRESENTAÇÃO DA ESPECIFICAÇÃO

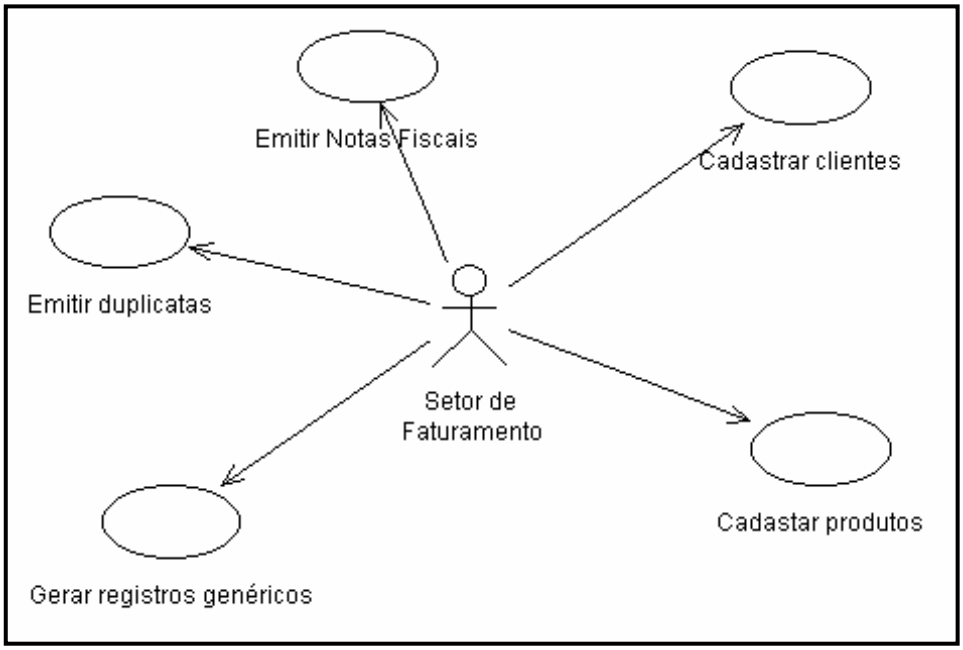
Na modelagem do protótipo foi utilizado análise orientado a objeto baseado em UML, através da ferramenta Rose da empresa Rational Inc. Foi desenvolvido a análise de requisitos, diagrama de classes, diagrama de seqüência e o *design*. Enquanto que no ambiente Oracle objeto relacional será utilizado análise orientado a objeto, no ambiente Oracle relacional versão 7 será utilizada a análise estruturada.

O protótipo serve para avaliação dos ambientes, para tanto foi desenvolvido um aplicativo da área financeira o qual conterá informações de um sistema de faturamento. Tais informações serão fornecidas pelo “ator externo”, que é o usuário do sistema, e o mesmo irá fazer simulações das rotinas de cadastro de informações de clientes, produtos, notas fiscais e duplicatas, ainda há uma rotina de gerar dados genéricos, tal rotina tem a função dar liberdado ao usuário o qual poderá optar por ocorrer um problema proprositalmente na execução da rotina ou não ocorrer o problema.

5.2.1 ANÁLISE DE REQUISITOS

O sistema utiliza funções básicas como cadastrar, emitir, gerar registros entre outros. Com essas funções foi modelado, conforme figura 10, o diagrama de *use-case* do sistema de faturamento.

Figura 10: Diagrama de *use-case*. Fase de análise de requisitos

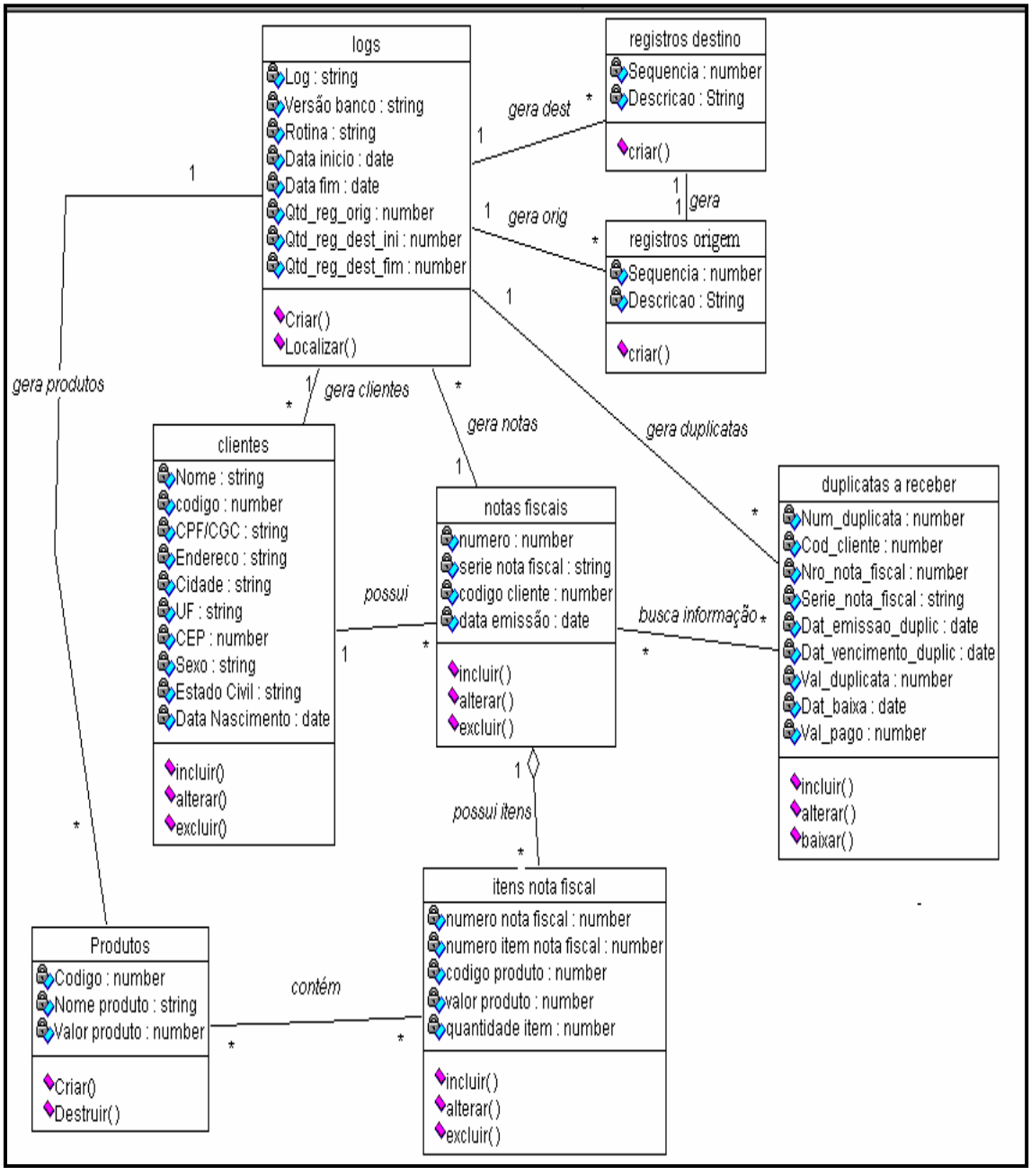


5.2.2 DIAGRAMA DE CLASSES

Após a confecção do diagrama de *use-case,* define-se o diagrama de classes, conforme figura 11. Esse diagrama deve ser livre em relação ao tipo de técnica a ser utilizada na

implementação do sistema, ou seja, deve ser desconsiderado os métodos, utilização de banco de dados, estrutura de mensagens entre objetos, entre outros. Nesse diagrama podemos observar como as classes se associam e se agregam. A classe denominada logs, é utilizada para criar um objeto sempre que houver alguma ação no aplicativo, como por exemplo, uma pesquisa de um determinado produto, a rotina irá gerar um objeto da classe logs com o tempo de acesso ao banco de dados utilizado na busca das informações do produto selecionado.

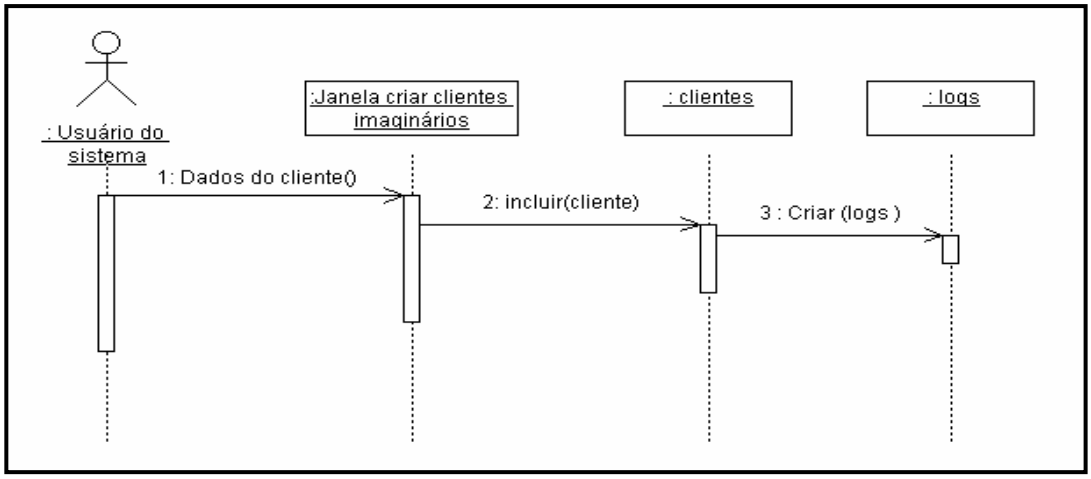
Figura 11: Diagrama de classes. Fase de análise.



5.2.3 DIAGRAMA DE SEQÜÊNCIA

Após a conclusão do diagrama de *uses-cases* e o diagrama de classes, devem ser elaborados as funções do sistema, e para que sejam modeladas essas funções é utilizado o diagrama de seqüência ou colaboração , conforme figura 12, para demonstrar a ordem cronológica das interações entre os objetos. Nesse momento são utilizadas idéias básicas de modelagem da interface do sistema como por exemplo as janelas.

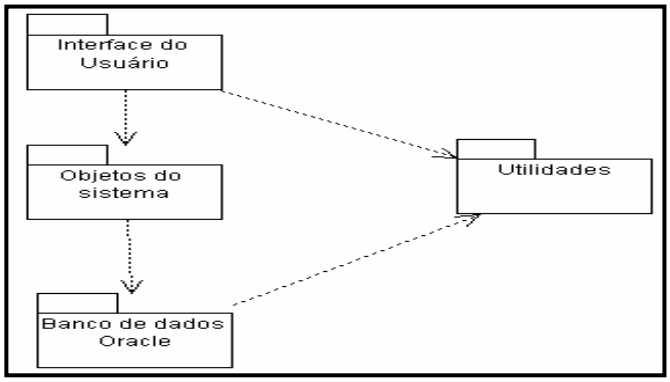
Figura 12: Diagrama de seqüência.



5.2.4 DESIGN

Nessa fase implementa-se melhoramentos no modelo e nas técnicas de como realmente cada função do sistema será criada. Conforme figura 13, pode-se identificar os “pacotes” (subsistemas), suas dependências e mecanismos de comunicação entre eles. O objetivo é criar uma arquitetura simples e clara, onde as dependências sejam poucas e que possam ser bidirecionais sempre que possível.

Figura 13: Fase de *Design*. Definição dos pacotes.



Após, desenvolvido os diagramas utilizando especificação UML, foi criado o desenho dos objetos na ferramenta CASE Oracle conforme figura 14, e após criou-se o modelo de entidade-relacionamento conforme figura 15, pois o sistema será implantado tanto no ambiente Oracle objeto relacional 8 como no ambiente Oracle relacional 7, sendo que o objetivo do trabalho é demonstrar como comporta-se uma rotina de notas fiscais do sistema de faturamento em ambos ambientes Oracle sempre objetivando a análise comparativa entre os mesmos.

Figura 14: Desenho dos objetos na ferramenta CASE Oracle

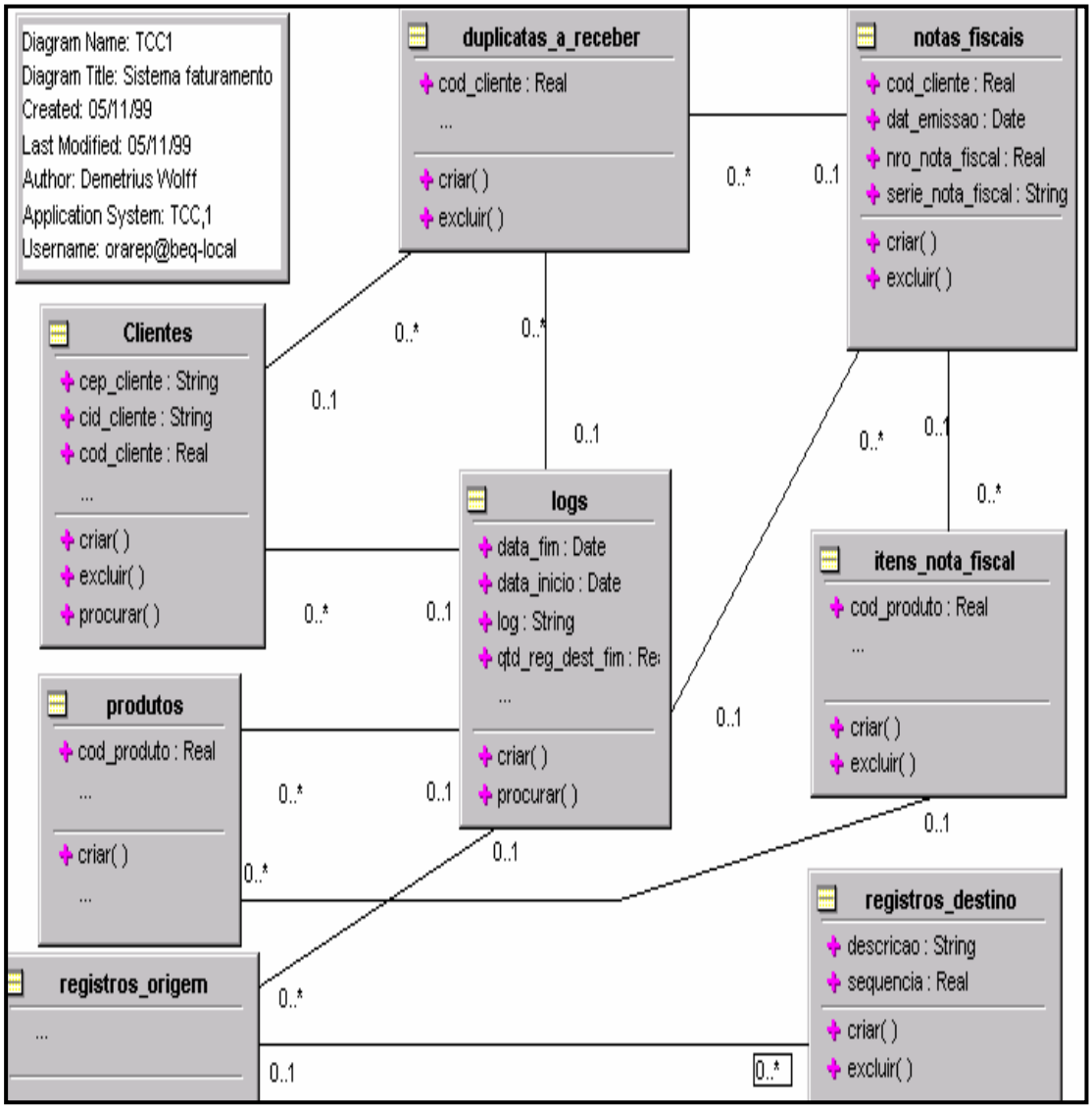
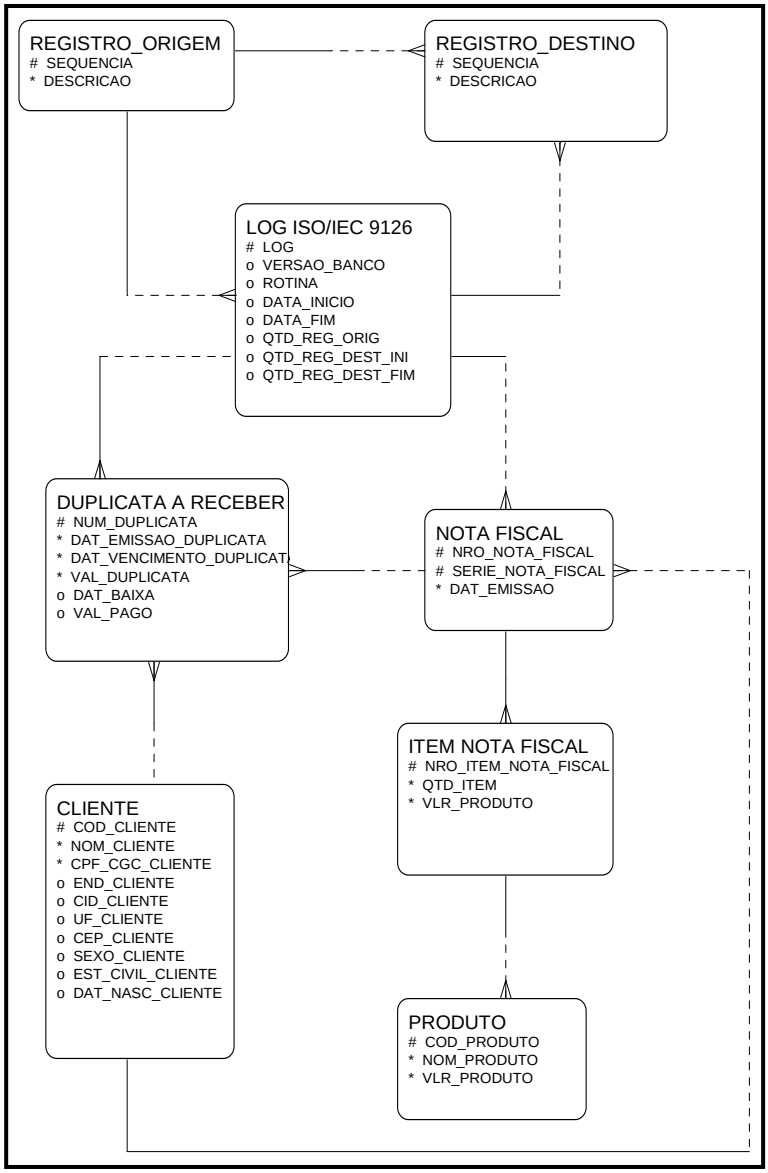


Figura 15: M.E.R. do protótipo.



As entidades foram normalizadas e de entidades passaram a ser denominadas de tabelas, as quais foram geradas tanto no ambiente Oracle relacional versão 7 quanto no ambiente Oracle objeto relacional versão 8.

5.2.5 DESCRIÇÃO DAS TABELAS

As tabelas foram descritas conforme quadro 5:

Quadro 5: Descrição das tabelas do protótipo.

Tabela / Entidade	**Campo / Atributo**	**Tipo**	**Tamanho**	**Descritivo**
Logs	Log	Alfanum.	06	Identificação
	Versao_banco	Alfanum.	80	Banco conectado
	Rotina	Alfanum.	50	Rotina de executada
	Data_inicio	Data		Data de inicio da rotina
	Data_fim	Data		Data finalização rotina
	Qtd_reg_orig	Numérico	12	Qtdade.reg.origem
	Qtd_reg_dest_ini	Numérico	12	Qtdade.reg. antes rotina
	Qtd_reg_dest_fim	Numérico	12	Qtdade.reg.depois rotina

Tabela / Entidade	**Campo / Atributo**	**Tipo**	**Tamanho**	**Descritivo**
Registros_origem	Sequencia	Numérico	12	Numero sequencial
	Descricao	Alfanum.	255	Descrição genérica

Tabela / Entidade	**Campo / Atributo**	**Tipo**	**Tamanho**	**Descritivo**
Registros_destino	Sequencia	Numérico	12	Numero sequencial

	Descricao	Alfanum.	255	Descrição genérica

Tabela / Entidade	Campo / Atributo	Tipo	Tamanho	Descritivo
Notas_fiscais	Nro_nota_fiscal	Numérico	10	Numero da nota fiscal
	Serie_nota_fiscal	Alfanum.	03	Série da nota fiscal
	Cod_Cliente	Numérico	05	Código do cliente
	Dat_emissao	Data		Data de emissão da nota

Tabela / Entidade	Campo / Atributo	Tipo	Tamanho	Descritivo
Itens_nota_fiscal	Nro_nota_fiscal	Numérico	10	Numero da nota fiscal
	Nro_item_nota	Numérico	10	Numero do ítem da nota
	Cod_produto	Numérico	10	Código do produto
	Vlr_produto	Numérico	15,2	Valor do produto
	Qtd_item	Numérico	05	Quantidade do ítem

Tabela / Entidade	Campo / Atributo	Tipo	Tamanho	Descritivo
Produtos	Cod_produto	Numérico	10	Código do produto
	Nom_produto	Alfanum.	40	Descrição do produto
	Vlr_produto	Numérico	15,2	Valor do produto

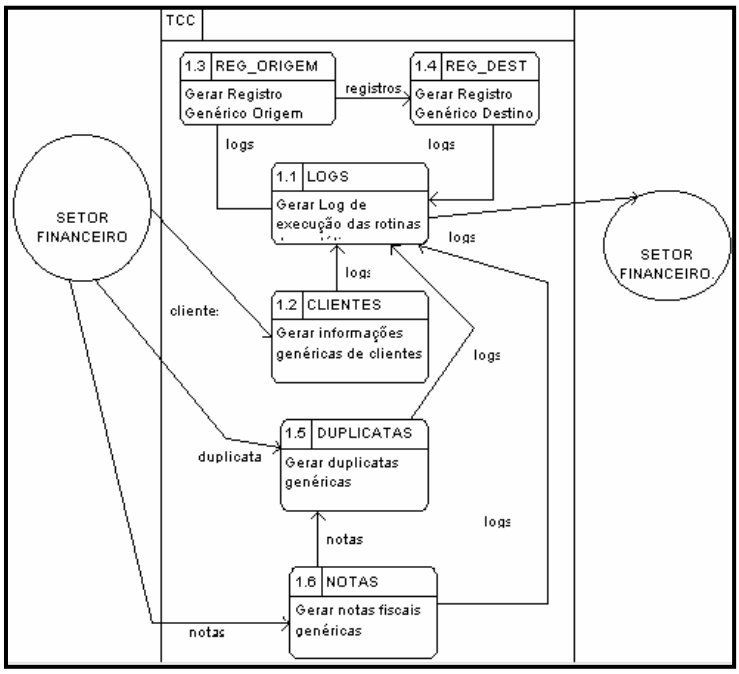
Tabela / Entidade	Campo / Atributo	Tipo	Tamanho	Descritivo
Clientes	Cod_cliente	Numérico	05	Código do cliente
	Nom_cliente	Alfanum.	40	Nome do cliente
	Cpf_cgc_cliente	Alfanum.	20	CPF/CGC do cliente
	End_cliente	Alfanum.	40	Endereço do cliente
	Cid_cliente	Alfanum.	20	Cidade que reside o cliente
	Uf_cliente	Alfanum.	02	Estado que reside o cliente
	Cep_cliente	Alfanum.	09	CEP da cidade
	Sexo_cliente	Alfanum.	01	Sexo do cliente
	Est_civil_cliente	Alfanum.	10	Estado civil do cliente
	Data_nasc_cliente	Data		Data de nascimento cliente

Tabela / Entidade	Campo / Atributo	Tipo	Tamanho	Descritivo
Duplicatas_a_recb	Num_duplicata	Number	10	Numero da duplicata
	Cod_cliente	Number	05	Código do cliente
	Nro_nota_fiscal	Number	10	Número da nota fiscal
	Serie_nota_fiscal	Alfanum.	03	Série da nota fiscal
	Dat_emissao_duplic	Data		Data de emissão duplicata

	Dat_vencimento_dp	Data		Data vencimento duplicata
	Val_duplicata	Number	15,2	Valor da duplicata
	Dat_baixa	Data		Data de baixa da duplicata
	Val_pago	Number	15,2	Valor pago da duplicata

5.2.6 DIAGRAMA DE FLUXO DE DADOS

Figura 16: D.F.D. do protótipo.



No diagrama de fluxo de dados do protótipo, conforme figura 16, estão demonstrados os processos que a rotina irá realizar. A rotina irá ser populada com informações genéricas, no processo 1.1, serão gravados todos procedimentos da rotina, ou seja, por exemplo: quando for gerada uma duplicata, será inserido um registro com informações de horário e quantidade de registros antes e depois da execução da rotina. No processo 1.2 serão gerados informações referente aos clientes e assim por diante nos demais processos. Ao final do processo, a rotina disponibilizará informações para que sejam gerados os gráficos comparativos.

5.2.7 ATRIBUIÇÃO DE VALORES PARA ANÁLISE COMPARATIVA

Para validação das características e subcaracterísticas da norma ISO/IEC 9126, sobre os ambientes analisados, serão executadas as mesmas rotinas do protótipo nos dois ambientes Oracle, o qual será adicionado um valor para a pontuação final de cada ambiente, conforme conceito, referente ao comportamento das execuções. A pontuação começa em zero. Ao final das execuções será verificado a pontuação de cada ambiente e assim chegando-se a uma resposta final. Serão utilizados como base os seguintes conceitos:

a) “Não satisfaz a característica requerida”, adiciona-se 0 (zero) a pontuação final ;

b) “Satisfaz parcialmente”, adiciona-se 5 (cinco) a pontuação final;

c) “Satisfaz totalmente”, adiciona-se 10 (dez) a pontuação final.

Entre os conceitos, poderão ocorrer variações, no caso de alguma análise do requisito avaliado não foi satisfeito totalmente, mas ficou acima de satisfação parcial, então atribui-se o valor que o avaliador determinar adequado.

5.2.8 CENÁRIO

Para a execução desta aplicação foi utilizado dois servidores (microcomputadores) distintos, conectados em rede, para cada versão do banco de dados conforme a seguir:

1) Microcomputador Pentium com processador de 200 Mhz, 128 Mbytes de memória RAM e com disco rígido de 4Gbytes controlado por placa SCSI. Nesse servidor encontra-se instalado o sistema operacional WindowsNT e o ambiente Oracle relacional versão 7.3.4 *worgroup*;

2) Microcomputador Pentium II com processador de 266 Mhz, 96 Mbytes de memória RAM e com disco rígido de 4 Gbytes controlado por placa SCSI. Nesse servidor encontra-se instalado o sistema operacional Linux e o ambiente Oracle objeto relacional versão 8.0.5 *standard edition*.

A rede (windows NT), possui 30 (trinta) estações conectadas, no momento da execução do protótipo somente a aplicação do protótipo estava sendo executada em ambos servidores. Em ambos bancos de dados, antes de executarem as rotinas do protótipo, as tabelas criadas encontram-se sem nenhum registros.

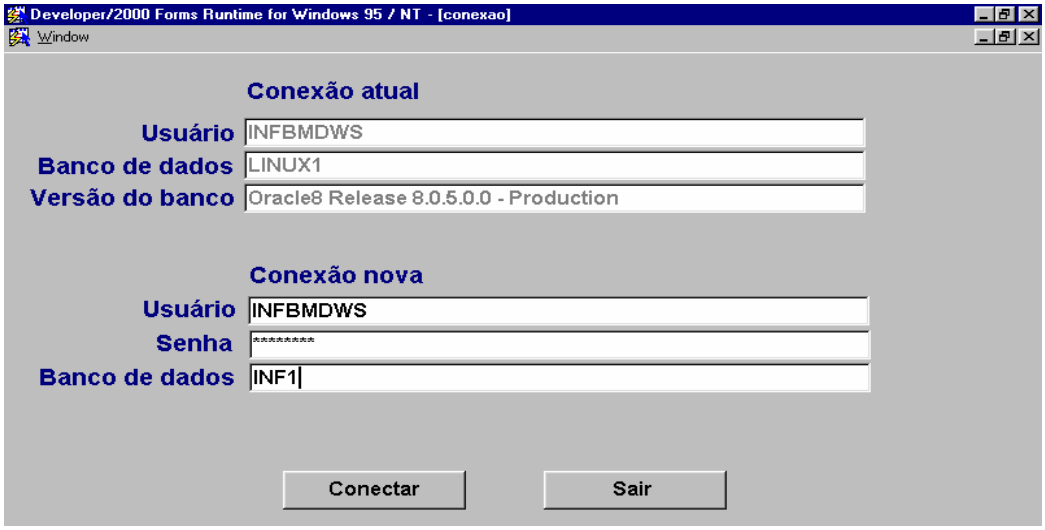
5.3 IMPLEMENTAÇÃO

Na implementação foi utilizado a ferramenta form's 4.5 de propriedade da empresa Oracle. O protótipo servirá como base para compararmos as duas versões.

Figura 17: Tela inicial do protótipo.

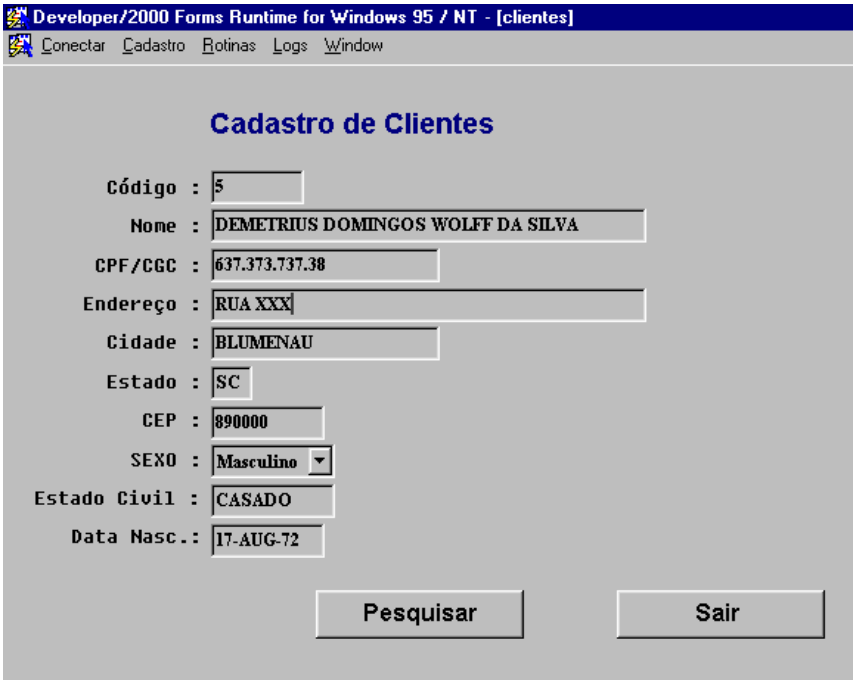
A figura 17 é a tela inicial do protótipo onde nela contém a versão que o usuário está conectado. A primeira opção do protótipo é a tela de conexão, conforme demonstrado na figura 18. Essa tela altera a conexão no banco de dados escolhido, ou seja não é necessário sair do protótipo para mudar de uma versão do banco de dados para outra versão. Após conectado no banco de dados, todas as informações são referente ao banco de dados informado na tela.

Figura 18: Tela de conexão.



A seguir, na opção cadastros, estão as telas de cadastro de clientes e cadastro de produtos, conforme descrito na figura 19.

Figura 19: Tela de cadastro.



Nessas telas, conforme figura 19, poderão ser realizadas inclusões, e no momento de realizar pesquisas foi implementado um botão denominado pesquisar, o qual, quando clicado,

irá gravar na tabela LOGS, informações referente ao tempo de acesso ao banco de dados, quantidade de registros antes e depois da pesquisa.

Figura 20: Tela rotina de geração de informações.

Na opção rotinas, descrita na figura 20, encontram-se as telas de criar clientes imaginários, criar produtos imaginários e criar registros origem/destino genéricos. Da mesma forma das telas de cadastro na figura 19, foi criado um botão de gerar, quando o mesmo for clicado, irá gravar na tabela de LOGS, informações referente ao tempo de acesso ao banco de dados, quantidade de registros antes e depois da pesquisa.

Figura 21: Tela de Logs de execuções das rotinas.

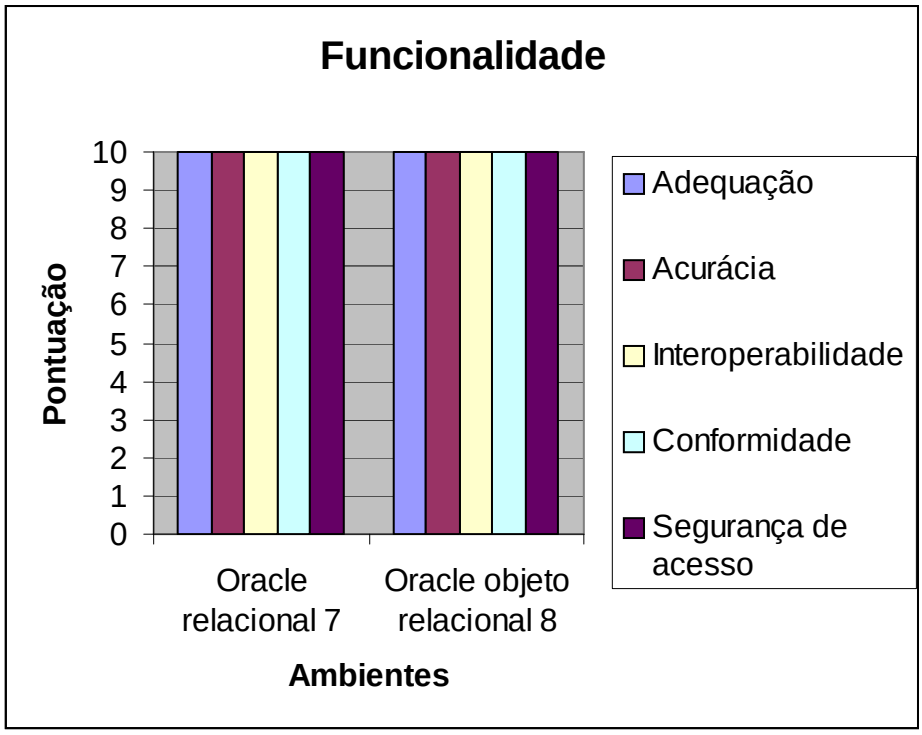
Na opção logs conforme figura 21, poderão ser consultados as execuções realizadas com a quantidade de registros e o tempo de execução.

Partindo do protótipo e das informações obtidas pelo mesmo, serão demonstrados os resultados, graficamente, das execuções das rotinas em cada ambiente Oracle das versões 7 e 8, de cada características conforme a norma ISO/IEC 9126, descritos no capitulo 4.5.1 Características e Subcaracterísticas da Qualidade.

5.3.1 ANÁLISE COMPARATIVA E INTERPRETAÇÃO DOS RESULTADOS

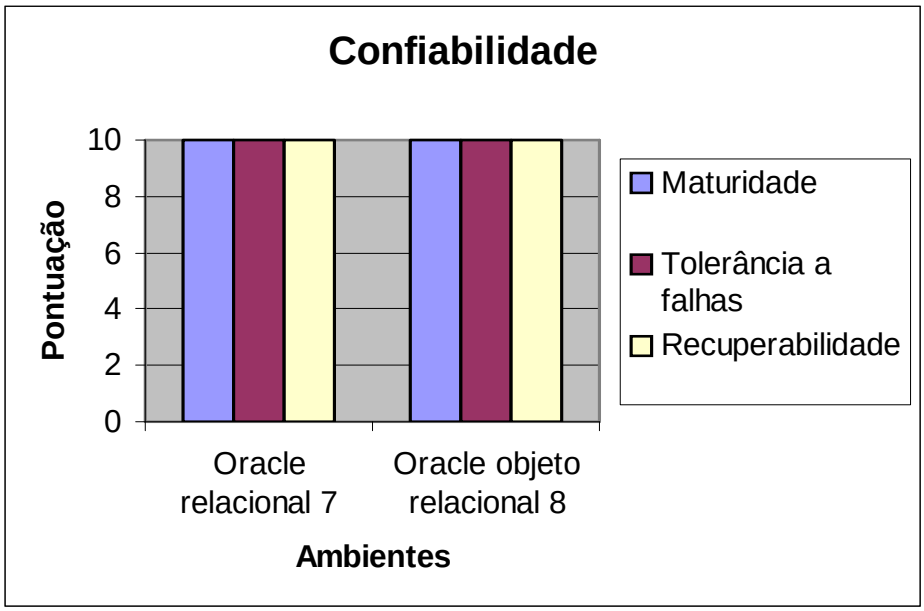
A partir desse momento, serão apresentas graficamente as análises comparativas e a interpretação dos resultados de cada característica e suas subcaracterísticas analisadas.

Figura 22: Característica funcionalidade da norma ISO/IEC 9126.



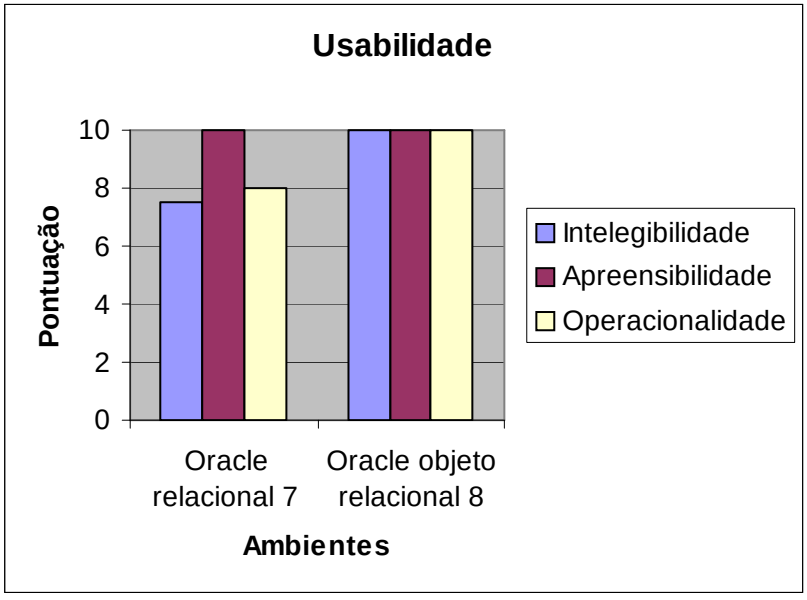
Na característica de funcionalidade, conforme demonstrado graficamente na figura 22, com os testes realizados, pode-se fazer a seguinte análise: os dois ambientes satisfazem totalmente essa característica.

Figura 23: Característica confiabilidade da norma ISO/IEC 9126.



Na característica confiabilidade, conforme demonstrado graficamente na figura 23, pôde-se verificar que tanto o ambiente Oracle relacional versão 7 como o ambiente Oracle objeto relacional versão 8 são totalmente confiáveis.

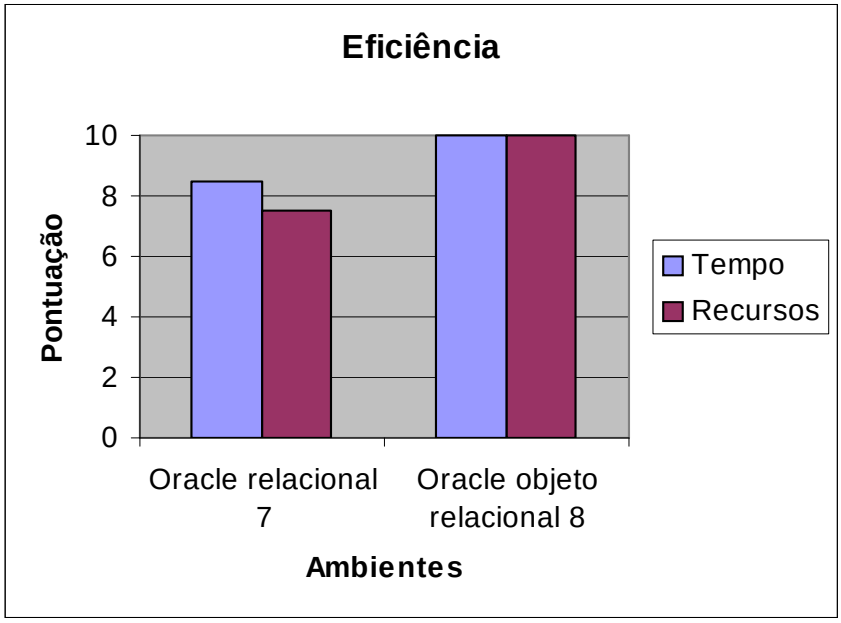
Figura 24: Característica usabilidade da norma ISO/IEC 9126.



Na característica usabilidade, conforme demonstrado graficamente na figura 24, verificou-se que no ambiente Oracle relacional versão 7 perante o ambiente Oracle objeto

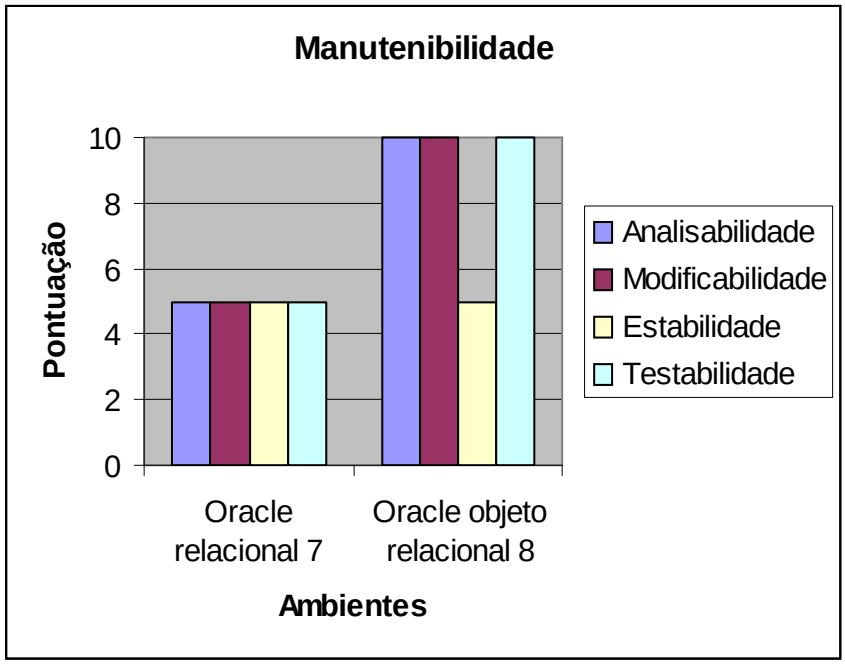
relacional 8, satisfez parcialmente a subcaracterística Intelegibilidade pelo fato que no ambiente Oracle objeto relacional 8 além de utilizar a conceito relacional, utiliza conceitos orientado a objetos. O ambiente Oracle relacional versão 7 satisfez parcialmente também a subcaracterística operacionalidade, pois comparando-o com o ambiente Oracle objeto relacional versão 8, esse traz novas facilidades de controle e operações. Pode-se observar este comparativo com a análise apresentada do capítulo 5.1 ao 5.2.6.

Figura 25: Característica eficiência da norma ISO/IEC 9126.



Na característica eficiência, conforme demonstrado graficamente na figura 25, podemos constatar que o ambiente Oracle objeto relacional versão 8, foi superior perante o ambiente Oracle relacional versão 7. Em relação ao tempo, podemos verificar no capítulo 7 o qual mostra os anexos referente aos resultados das execuções, e no recurso utilizado, podemos verificar que o equipamento utilizado para ambos ambientes foi praticamente o mesmo diferenciando somente no sistema operacional e na quantidade de memória conforme capitulo 5.2.8 o qual refere-se aos ambientes de hardware e software.

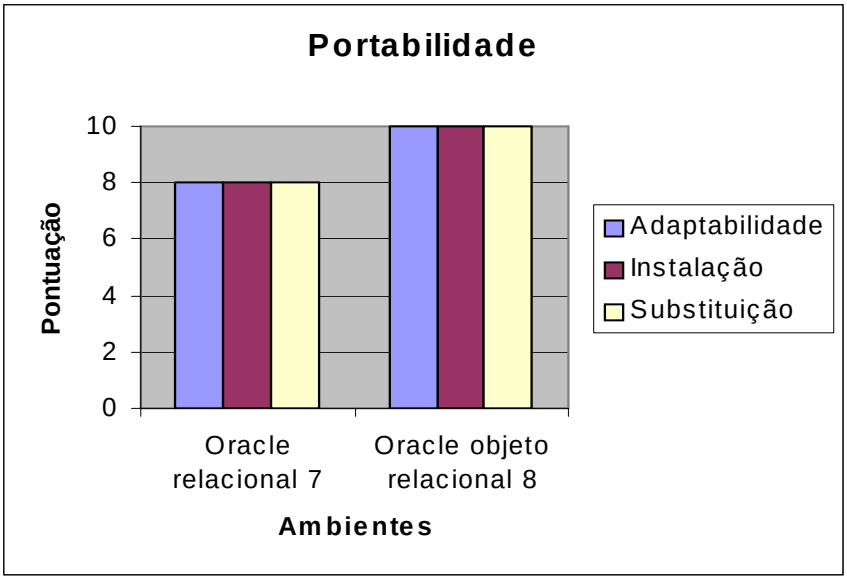
Figura 26: Característica manutenibilidade da norma ISO/IEC 9126.



Na característica manutenibilidade, conforme demonstrado graficamente na figura 26, podemos constatar que o ambiente Oracle objeto relacional versão 8, foi superior perante o ambiente Oracle relacional versão 7, pelo fato que no ambiente Oracle objeto relacional versão 8 comparando-se ao ambiente Oracle relacional versão 7 é bem mais simples de analisar o sistema desenvolvido pela análise orientado a objeto, e em decorrência disso, é mais simples modificá-lo e testá-lo.

Porém chama-se a atenção na subcaracterística estabilidade pelo fato de que quando são realizadas alterações de versões, tanto no ambiente relacional versão 7 como no ambiente objeto relacional versão 8, há riscos de uma versão não ser compatível com a outra, como por exemplo, uma versão não está preparada para aceitar um certo tipo de dado.

Figura 27: Característica portabilidade da norma ISO/IEC 9126.



Na característica portabilidade, conforme demonstrado graficamente na figura 27, foram entrevistados na empresa Heller Brasil Tecnologia o Sr. Jefferson Plebianca, responsável pelo setor de instalação de hardware e software, e na FURB o Sr. Vandeir Eduardo, funcionário do Laboratório do Protem, o qual instalou os ambientes Oracle. Verificou-se que em ambas organizações foi mais fácil a adaptação, instalação e substituição do ambiente Oracle objeto relacional versão 8 perante o ambiente Oracle relacional versão 7, pelo fato de haver entre outras facilidades, um número maior de “assistentes de instalação” que servem para o auxílio da instalação de software. O Sr. Vandeir ressaltou ainda que na plataforma *Windows* foi bem mais fácil instalar, comparando-se a outras plataformas por ele instaladas, e que no caso da instalação do Oracle objeto relacional versão 8 existe o instalador universal, que suporta a linguagem Java, o qual executa em qualquer plataforma.

6 CONCLUSÕES

Analisando os resultados obtidos com as comparações realizadas nos ambientes Oracle relacional versão 7 e Oracle objeto relacional versão 8, conclui-se que o ambiente Oracle objeto relacional versão 8 é superior ao ambiente Oracle relacional versão 7, principalmente os itens de usabilidade, eficiência, manutenibilidade e portabilidade. Com isso, vindo a comprovar-se que as divulgações por consultores, livros, artigos e outros, estão condizentes ao afirmarem que o ambiente Oracle objeto relacional versão 8 além de ser superior às versões anteriores, inclui o conceito de orientação a objeto, comprovados na análise da especificação do sistema com testes realizados neste trabalho.

Os padrões da norma ISO/IEC 9126 foram fundamentais para auxilio na avaliação, pois foi através das características e subcaracterísticas dessa norma que avaliei os ambientes Oracle relacional versão 7 e Oracle objeto relacional versão 8 e conclui esse trabalho .

Em relação ao mercado, esse trabalho contribui no auxilio da aquisição ou não dos ambientes avaliados, e aos desenvolvedores de software demonstra uma inicialização de como desenvolver sistemas utilizando análise orientado a objeto.

Em particular, esse trabalho de conclusão de curso, ajudou-me a visualizar novas técnicas de análise de sistemas como a UML, com o crescente demanda no mercado de empresas que desejam adquirir o banco de dados Oracle mostrou-me oportunidades de desenvolvimento de sistemas utilizando orientação a objeto. Atualmente na empresa onde sou colaborador, Heller Brasil Tecnologia, existe um espaço aonde podem ser expostos assuntos técnicos ou novidades na área de informática, senti-me motivado a demonstrar essa análise comparativa para que também todos os colaboradores e interessados possam estar atualizados referente a esse novo ambiente disponibilizado pela empresa Oracle.

6.1 DIFICULDADES DURANTE ESSE TRABALHO

Foram encontradas dificuldades nesse trabalho, pois há exigências na instalação do ambiente Oracle objeto relacional versão 8, onde a principal dificuldade é que para que sejam implementadas aplicações com orientação a objeto deve ser instalada a versão integral do banco de dados Oracle objeto relacional versão 8 *Enterprise Edition*, e para isso o

equipamento deve ser uma máquina com bons recursos de hardware, ou seja, processador rápido, mínimo de 128 Megabytes de memória RAM e disco rígido de 2 Gigabytes.

6.2 SUGESTÕES

Como sugestões para próximos trabalhos, recomendo :

1) realizar a análise comparativa entre bancos de dados de outros fabricantes de versões atuais com versões anteriores;

2) implementar um sistema com orientação a objeto e implementar um aplicativo para fazer o mapeamento do modelo objeto para o modelo relacional;

3) utilização de outras normas para realização da análise comparativa entre softwares.

7 ANEXOS

7.1 ROTINAS DO PROTÓTIPO

Nas rotinas do protótipo foi utilizada a linguagem PL/SQL, conforme demonstração de algumas rotinas do protótipo:

7.1.1 ROTINA DE PESQUISA DE CLIENTES

```
DECLARE
  AUX_COD_CLIENTE    NUMBER(5);
  AUX_CONT           NUMBER;
  AUX_LOG            VARCHAR2(6);
  AUX_VERSAO_BANCO   VARCHAR2(80);
  AUX_QTD_REG        NUMBER(12);
BEGIN
  SELECT NVL(MAX(LOG),0) + 1
    INTO AUX_LOG
    FROM LOGS;

  SELECT BANNER
    INTO AUX_VERSAO_BANCO
    FROM V$VERSION
   WHERE ROWNUM = 1;

  SELECT COUNT(*)
    INTO AUX_QTD_REG
    FROM CLIENTES;

  SELECT NVL(MAX(COD_CLIENTE),0) + 1
    INTO AUX_COD_CLIENTE
    FROM CLIENTES;

  INSERT INTO LOGS (LOG,VERSAO_BANCO,ROTINA,DATA_INICIO,QTD_REG_ORIG,QTD_REG_DEST_INI)
           VALUES                                              (AUX_LOG,AUX_VERSAO_BANCO,'PESQUISA
CLIENTES',SYSDATE,AUX_QTD_REG,AUX_QTD_REG);

  EXECUTE_QUERY;

  SELECT COUNT(*)
    INTO AUX_QTD_REG
    FROM CLIENTES;

  UPDATE LOGS SET DATA_FIM         = SYSDATE
                 ,QTD_REG_DEST_FIM = AUX_QTD_REG
     WHERE LOG = AUX_LOG;

  :SYSTEM.MESSAGE_LEVEL := 20;
  COMMIT;
  :SYSTEM.MESSAGE_LEVEL := 0;
  MESSAGE('TERMINO DA PESQUISA DE CLIENTES ',NO_ACKNOWLEDGE);
  BELL;
  SYNCHRONIZE;
END;
```

7.1.2 ROTINA DE GERAÇÃO DAS DUPLICATAS

```
DECLARE
  AUX_NUM_DUPLICATA          NUMBER(10);
  AUX_CONT                   NUMBER;
```

```
  AUX_CONT_REG               NUMBER;
  AUX_LOG                    VARCHAR2(6);
  AUX_VERSAO_BANCO           VARCHAR2(80);
  AUX_QTD_REG                NUMBER(12);

  CURSOR CURSOR1 IS
    SELECT A.NRO_NOTA_FISCAL
          ,A.SERIE_NOTA_FISCAL
          ,A.COD_CLIENTE
          ,SUM(B.VLR_PRODUTO) * SUM(B.QTD_ITEM) VAL_DUPLICATA
      FROM NOTAS_FISCAIS     A
          ,ITENS_NOTA_FISCAL B
     WHERE A.NRO_NOTA_FISCAL = B.NRO_NOTA_FISCAL
    GROUP BY A.NRO_NOTA_FISCAL
            ,A.SERIE_NOTA_FISCAL
            ,A.COD_CLIENTE;

BEGIN
  IF :DUPLICATAS_A_RECEBER.QUANTIDADE_DUPLICATAS IS NULL THEN
    MESSAGE('QUANTIDADE DE DUPLICATAS NÃO PODE SER NULO');
    BELL;
    RAISE FORM_TRIGGER_FAILURE;
  END IF;

  IF :DUPLICATAS_A_RECEBER.QUANTIDADE_NOTAS IS NULL THEN
    MESSAGE('QUANTIDADE DE NOTAS FISCAIS NÃO PODE SER NULO');
    BELL;
    RAISE FORM_TRIGGER_FAILURE;
  END IF;

  IF :DUPLICATAS_A_RECEBER.DATA_EMISSAO IS NULL THEN
    MESSAGE('DATA DA EMISSÃO DA DUPLICATA NÃO PODE SER NULO');
    BELL;
    RAISE FORM_TRIGGER_FAILURE;
  END IF;

  IF :DUPLICATAS_A_RECEBER.PERIODO_VENCIMENTO IS NULL THEN
    MESSAGE('O PERIODO DO VENCIMENTO NÃO PODE SER NULO');
    BELL;
    RAISE FORM_TRIGGER_FAILURE;
  END IF;

  SELECT NVL(MAX(LOG),0) + 1
    INTO AUX_LOG
    FROM LOGS;

  SELECT BANNER
    INTO AUX_VERSAO_BANCO
    FROM V$VERSION
   WHERE ROWNUM = 1;

  SELECT COUNT(*)
    INTO AUX_QTD_REG
    FROM DUPLICATAS_A_RECEBER;

  SELECT NVL(MAX(NUM_DUPLICATA),0)
    INTO AUX_NUM_DUPLICATA
    FROM DUPLICATAS_A_RECEBER;

  INSERT INTO LOGS (LOG,VERSAO_BANCO,ROTINA,DATA_INICIO,QTD_REG_ORIG,QTD_REG_DEST_INI)
            VALUES                                              (AUX_LOG,AUX_VERSAO_BANCO,'GERA
DUPLICATAS',SYSDATE,AUX_QTD_REG,AUX_QTD_REG);

  AUX_CONT_REG := 0;
  FOR C1 IN CURSOR1 LOOP
    AUX_CONT_REG := AUX_CONT_REG + 1;
    EXIT WHEN AUX_CONT_REG > NVL(:DUPLICATAS_A_RECEBER.QUANTIDADE_NOTAS,0);
    FOR AUX_CONT IN 1 .. :DUPLICATAS_A_RECEBER.QUANTIDADE_DUPLICATAS LOOP
      AUX_NUM_DUPLICATA := AUX_NUM_DUPLICATA + 1;
      MESSAGE('GERANDO        DUPLICATA        '||TO_CHAR(AUX_NUM_DUPLICATA)||'        CLIENTE
'||TO_CHAR(C1.COD_CLIENTE)||' NOTA FISCAL '||TO_CHAR(C1.NRO_NOTA_FISCAL),NO_ACKNOWLEDGE);
      SYNCHRONIZE;
```

```
      BEGIN
        INSERT INTO DUPLICATAS_A_RECEBER(NUM_DUPLICATA
                                        ,COD_CLIENTE
                                        ,NRO_NOTA_FISCAL
                                        ,SERIE_NOTA_FISCAL
                                        ,DAT_EMISSAO_DUPLICATA
                                        ,DAT_VENCIMENTO_DUPLICATA
                                        ,VAL_DUPLICATA)
                                  VALUES(AUX_NUM_DUPLICATA
                                        ,C1.COD_CLIENTE
                                        ,C1.NRO_NOTA_FISCAL
                                        ,C1.SERIE_NOTA_FISCAL
                                        ,:DUPLICATAS_A_RECEBER.DATA_EMISSAO
                                        ,:DUPLICATAS_A_RECEBER.DATA_EMISSAO                    +
(:DUPLICATAS_A_RECEBER.PERIODO_VENCIMENTO * AUX_CONT)
                                        ,C1.VAL_DUPLICATA);
      EXCEPTION
        WHEN OTHERS THEN
          MESSAGE('PROBLEMAS NA GERAÇÃO DE DUPLICATAS - ERRO '||SQLERRM);
          BELL;
          RAISE FORM_TRIGGER_FAILURE;
      END;
    END LOOP;
  END LOOP;
  SELECT COUNT(*)
    INTO AUX_QTD_REG
    FROM PRODUTOS;

  UPDATE LOGS SET DATA_FIM         = SYSDATE
                 ,QTD_REG_DEST_FIM = AUX_QTD_REG
     WHERE LOG = AUX_LOG;

  :SYSTEM.MESSAGE_LEVEL := 20;
  COMMIT;
  :SYSTEM.MESSAGE_LEVEL := 0;
  MESSAGE('TERMINO DA GERAÇÃO DE DUPLICATAS ',NO_ACKNOWLEDGE);
  BELL;
  SYNCHRONIZE;
END;
```

7.2 RESULTADOS DAS EXECUÇÕES

```
SQL> SELECT BANNER FROM V$VERSION;

BANNER
----------------------------------------------------------------
ORACLE7 SERVER RELEASE 7.3.2.3.8 - PRODUCTION RELEASE
PL/SQL RELEASE 2.3.2.3.8 - PRODUCTION
CORE VERSION 3.5.2.0.0 - PRODUCTION
TNS FOR 32-BIT WINDOWS: VERSION 2.3.2.1.0 - PRODUCTION
NLSRTL VERSION 3.2.2.0.0 - PRODUCTION

SQL> SELECT LOG
  2        ,SUBSTR(VERSAO_BANCO,1,7)                    BANCO
  3        ,SUBSTR(ROTINA,1,20)                         ROTINA
  4        ,TO_NUMBER(TO_CHAR(DATA_FIM,'HH24MISS'))
          - TO_NUMBER(TO_CHAR(DATA_INICIO,'HH24MISS'))  TEMPO
  5        ,QTD_REG_ORIG
  6        ,QTD_REG_DEST_INI
  7        ,QTD_REG_DEST_FIM
  8  FROM LOGS;

LOG    BANCO   ROTINA                   TEMPO QTD_REG_ORIG QTD_REG_DEST_INI QTD_REG_DEST_FIM
------ ------- -------------------- --------- ------------ ---------------- ----------------
1      ORACLE7 GERA CLIENTES              186            0                0            10001
2      ORACLE7 GERA CLIENTES              851        10001            10001            60002
3      ORACLE7 GERA PRODUTOS              360            0                0            20001
4      ORACLE7 GERA PRODUTOS              688        20001            20001            65002
5      ORACLE7 GERA NOTAS                5288            0                0            65002
6      ORACLE7 GERA DUPLICATAS             10            0                0            65002
```

```
7      ORACLE7 GERA DUPLICATAS            12          150             150            65002
8      ORACLE7 GERA REGISTROS GENER       19            0               0             1000
9      ORACLE7 GERA REGISTROS GENER       60         1000            1000             2000
10     ORACLE7 PESQUISA CLIENTES         117        60002           60002            65002
10     ORACLE7 PESQUISA CLIENTES          48        60002           60002            65002
10     ORACLE7 PESQUISA PRODUTOS          14        65002           65002            65002
10     ORACLE7 PESQUISA PRODUTOS           0        65002           65002            65002

13 ROWS SELECTED.

SQL> CONN INFBMDWS@LINUX2
CONNECTED.
SQL> SELECT BANNER FROM V$VERSION;

BANNER
----------------------------------------------------------------
ORACLE8 RELEASE 8.0.5.0.0 - PRODUCTION
PL/SQL RELEASE 8.0.5.0.0 - PRODUCTION
CORE VERSION 4.0.5.0.0 - PRODUCTION
TNS FOR LINUX: VERSION 8.0.5.0.0 - PRODUCTION
NLSRTL VERSION 3.3.2.0.0 - PRODUCTION

SQL> SELECT LOG
  2        ,SUBSTR(VERSAO_BANCO,1,7)                    BANCO
  3        ,SUBSTR(ROTINA,1,20)                         ROTINA
  4        ,TO_NUMBER(TO_CHAR(DATA_FIM,'HH24MISS'))
          - TO_NUMBER(TO_CHAR(DATA_INICIO,'HH24MISS'))  TEMPO
  5        ,QTD_REG_ORIG
  6        ,QTD_REG_DEST_INI
  7        ,QTD_REG_DEST_FIM
  8  FROM LOGS;

LOG    BANCO   ROTINA                   TEMPO QTD_REG_ORIG QTD_REG_DEST_INI QTD_REG_DEST_FIM
------ ------- -------------------- --------- ------------ ---------------- ----------------
1      ORACLE8 GERA CLIENTES              180            0                0            10001
2      ORACLE8 GERA CLIENTES              800        10001            10001            60002
3      ORACLE8 GERA PRODUTOS              376            0                0            20001
4      ORACLE8 GERA PRODUTOS              812        20001            20001            65002
5      ORACLE8 GERA NOTAS                1525            0                0            65002
6      ORACLE8 GERA DUPLICATAS              6            0                0            65002
7      ORACLE8 GERA DUPLICATAS             49          150              150            65002
8      ORACLE8 GERA REGISTROS GENER        22            0                0             1000
9      ORACLE8 GERA REGISTROS GENER        62         1000             1000             2000
10     ORACLE8 PESQUISA CLIENTES           74        60002            60002            65002
10     ORACLE8 PESQUISA CLIENTES           23        60002            60002            65002
10     ORACLE8 PESQUISA PRODUTOS            9        65002            65002            65002
10     ORACLE8 PESQUISA PRODUTOS            0        65002            65002            65002

13 ROWS SELECTED.
```

REFERÊNCIAS BIBLIOGRÁFICAS

[ARA95] ARAÚJO, Sérgio P.C. **Uma abordagem de apoio à reutilização de componentes em ferramentas Oracle.** Blumenau, 1999. Trabalho de conclusão de curso (Bacharelado em Ciências da Computação) Centro de Ciências Exatas e Naturais, FURB.

[BAR94] BARBIERI, Carlos. **Modelagem e Administração de Dados.** Rio de Janeiro : Infobook, 1994.

[BAR98] BARROS, Pablo Fernando do Rêgo. **UML – Linguagem de Modelagem Unificada.** Aracaju, 1998. Trabalho de conclusão de curso (Tecnólogo em Processamento de Dados), Centro de Ciências Formais e Tecnológicas – Universidade Tiradentes.

[BIN94] BINDER, Fábio Vinícios. **Sistemas de apoio à decisão.** São Paulo : Érica, 1994.

[CER95] CERÍCOLA, Vicente Oswald. **ORACLE - Banco de dados relacional e distribuído. Ferramentas para desenvolvimento.** São Paulo : Makron - McGraw-Hill, 1995.

[CHA99] CHAVES, Eduardo O.C. **Sistemas de informação e sistemas de apoio à decisão.** 1999. Endereço eletrônico: http://www.puccamp.br/~labi/staff/nov-1994.html#1.

[CHE77] CHEN, Peter. **Modelagem de dados. Relacionamento para projeto lógico.** São Paulo : Makron Books, 1977.

[CHU83] CHU, Shao Yong. **Banco de dados. Organização, Sistemas e Administração.** São Paulo : Atlas, 1983.

[COA92] COAD, Peter. **Analise baseada em objetos** Rio de Janeiro: Campus, 1992.

[COL96] COLEMAN, Derek. **Desenvolvimento orientado a objetos: o método fusion.** Rio de Janeiro: Campus, 1996.

[COR99] CORTES, Sérgio da Costa. **Banco de dados: fundamentos, projeto e implementação.** Rio de Janeiro: LTC, 1999.

[COU97] COUGO, Paulo Sérgio. **Modelagem conceitual e projeto de banco de dados.** Rio de Janeiro: Campus, 1997.

[DAL98] DALFOVO, Oscar. **Desenho de um Modelo de Sistemas de Informação.** Blumenau, 1998. Dissertação de mestrado em Administração de Negócios, Centro de Ciências Sociais e Aplicadas - FURB.

[DAT91] DATE, C.J. **Introdução a sistemas de banco de dados.** Rio de Janeiro: Campos, 1991.

[FAR89] FARRER, Harry. **Algoritmos Estruturados.** Rio de Janeiro: LTC, 1989.

[FUR98] FURLAN, José Davi. **Modelagem de Objetos através da UML.** São Paulo: Makron Books, 1998.

[JUN99] JÚNIOR, José Barreto. **Qualidade de Software.** 1999, Endereço eletrônico : http://www.barreto.com.br/qualidade/index.html.

[GRI94] EQUIPE GRIFO. **Iniciando os conceitos da qualidade total.** São Paulo: Pioneira, 1994.

[KER94] KERN, Vinicius Medina. **Banco de dados relacionais teoria e prática de projeto.** São Paulo: Érica, 1994.

[KHO94] KHOSHAFIAN, Setrag. **Banco de dados orientado a objetivos.** Rio de Janeiro: Infobook, 1994.

[KOR95] KORTH, Henry F. **Sistema de banco de dados.** São Paulo: Makron Books, 1995.

[MAR94] MARTIN, James. **Princípios de análise e projeto baseados em objetos.** Rio de Janeiro: Campus, 1994.

[MCC93] McCLURE, Carma. **The three Rs of software automation: re-engineering, repository, reusability.** New Jersey : Pretince Hall, 1993.

[MOR95] MORAIS, Rinaldo de Oliveira. **Oracle 7 Server: Conceitos básicos.** São Paulo: Érica, 1995.

[NET88] NETO, Acácio Feliciano. **Engenharia da informação: metodologia, técnicas e ferramentas.** São Paulo: McGraw-Hill, 1988.

[NOL77] NOLAN, Richard L. **Management accouting and control of data processing.** National Association of Accountants, 1977.

[OLI92] OLIVEIRA, Djalma de Pinto Rebouças de. **Sistema de informações gerenciais, táticas, operacionais.** São Paulo : Atlas, 1992.

[PIS96] PISKE, Rosilene. **Protótipo de um sistema de avaliação de qualidade softwares de folha de pagamento.** Blumenau, 1996. Trabalho de conclusão de curso (Bacharel em Ciências da Computação), Centro de Ciências Exatas e Naturais - FURB.

[PRA94] PRATES, Maurício. **Conceituação de sistemas de informação do ponto de vista do gerenciamento.** Revista do Instituto de Informática, PUC-CAMP, março/setembro, 1994.

[RAM99] RAMALHO, José Antônio. **Oracle 8i.** São Paulo : Berkeley Brasil, 1999.

[ROC95] ROCHA, Ana Regina Cavalcanti da. **Controle de qualidade de software.** Rio de Janeiro, 1995. Coordenação de programas de pós graduação em engenharia, programa de engenharia de sistemas e computação, Universidade Federal do Rio de Janeiro - COPPE.

[SAL95] SALEMI, Joe. **Banco de dados cliente/servidor.** Rio de Janeiro: IBPI, 1995.

[SET86] SETZER, Waldemar W. **Banco de dados.** São Paulo: Edgard Blücher, 1986.

[SHI93] SHILLER, Larry. **Excelência em software.** São Paulo : Makron, 1993.

[TIR98] TIRELLI, Aldir. **Análise comparativa entre os bancos de dados Adabas, Dataflex, Oracle e Progress**. Blumenau, 1998. Trabalho de Conclusão de Curso (Bacharelado em Ciências da Computação) Centro de Ciências Exatas e Naturais, FURB.

[WIN93] WINBLAD, Ann L. **Software orientado a objeto.** São Paulo : Makron Books, 1993.

[YOU95] YORDON, Edward. **Declínio e queda dos analistas e programadores.** São Paulo : Makron Books, 1995.