

**Monografia do Curso de Pós-Graduação
Especialização em Tecnologia em Desenvolvimento para Web.**

Arquitetura Oracle para Web

Claudia Maria Betteti Meirelles Gouveia

Arquitetura Oracle para Web

Monografia apresentada ao departamento de Informática da UEM como parte dos requisitos para obtenção do título de especialista em Tecnologia em Desenvolvimento para WEB.

Orientador: Edmundo Sérgio Spoto

Departamento de Informática
Universidade Estadual de Maringá

Maringá, janeiro de 2004

Agradecimentos

Ao Professor Cesar, pelo apoio na elaboração deste trabalho e a disponibilidade em repassar seus conhecimentos.

Ao Professor Edmundo, um dos idealizadores do curso de Especialização, e que desde o início da minha formação acadêmica repassou seus conhecimentos e me contemplou com sua amizade.

A Luana e Vitória, minhas filhas, que durante as aulas e elaboração deste trabalho ficaram sem minha atenção, e que são minha alegria de viver.

Ao Bill, que me ajudou com as minhas filhas durante o período de aulas.

Aos meus amigos do curso de Especialização, em especial, ao Henrique que muito me ajudou durante o curso.

Aos amigos do Departamento de informática – UEM, e em especial ao professor Wesley pelo trabalho frente à coordenação do curso de Especialização.

Aos professores do curso de Especialização que sempre estiveram a disposição dos alunos para transmitir seus conhecimentos.

Resumo

Este trabalho descreve os conceitos básicos do ambiente Web. Esta monografia apresenta a arquitetura Oracle para *Internet*. O conceito de arquitetura de três camadas é detalhado, suas características e as vantagens na utilização desta arquitetura. São apresentadas as características principais, bem como os padrões suportados e as vantagens do *Oracle Internet Application Server*. É feita uma breve referência sobre o que o *Web Application Server* suporta em relação a segurança dos aplicativos na *Web*.

É mostrado o principal componente da arquitetura da *Oracle* para *Internet* o banco de dados *Oracle9i*. São descritos seus recursos, as vantagens e a série de incrementos ocorridos sobre as versões anteriores. É feita uma breve exposição do ambiente intuitivo de modelagem *Oracle Designer*, descrevendo seus recursos. É abordada detalhadamente a ferramenta para desenvolvimento de aplicativos para ambiente *Web*, o *Oracle Forms Developer*, e o *Forms Services* que é o conjunto de programas e serviços que facilitam a execução destes aplicativos. É também abordado detalhadamente o *Oracle Reports Developer* que é a ferramenta para definição, extração e geração de relatórios. É apresentado o *Oracle9i Development Suite* que oferece um ambiente integrado que combina ferramentas de desenvolvimento de aplicativos e de *business intelligence* e o *Jdeveloper* para desenvolvimento de aplicações *Java* que é o principal componente deste ambiente.

Palavras Chaves: *Banco de Dados, Internet, Aplicações Web*

Abstract

This work describes the basic concepts of the atmosphere Web. This monograph presents the architecture Oracle for Internet. The concept of architecture of three layers is detailed, its characteristics and the advantages in the use of this architecture. It is presented the main characteristics, the supported patterns and the advantages of the Oracle Internet Application Server. It is made a brief reference on the one that him Web Application Server supports in relation to safety of the applications in the Web.

The main component of the architecture of Oracle is shown for Internet the database Oracle9i. Its resources, the advantages and the series of increments happened on the previous versions are described. It is made a brief exhibition of the intuitive atmosphere of modeling Oracle Designer, describing its resources. It is approached the tool in full detail for development of applications for ambient Web, Oracle Forms Developer, and Forms Services that it is the group of programs and services that facilitate the execution of these applications. It is also approached in full detail Oracle Reports Developer that is the tool for definition, extraction and generation of reports. It is presented Oracle9i Development Suite that offers an integrated atmosphere that combines tools of development of applications and of business intelligence and Jdeveloper for development of applications Java that is the main component of this sets.

Palavras Chaves: Database, *Internet*, *Web Application*

Sumário

1. Introdução.....	1
1.1 Objetivos	1
1.2 Organização.....	2
2. Revisão Bibliográficas	3
2.1. World Wide Web	3
2.2 Arquitetura Cliente/Servidor.....	4
2.3 Arquitetura Multi-camadas	5
2.3.1 Algumas das vantagens da utilização da arquitetura de 3 camadas	6
2.4 Segurança	7
2.5 Banco de Dados	7
2.6 GUI – Interface Gráfica do Usuário	9
2.7 SQL – Structure Query Language	9
2.8 PL/SQL – Procedural Language for SQL	10
2.8.1 <i>Procedures</i> e Funções	11
2.8.2 <i>Package</i>	12
2.8.3 <i>Database Triggers</i>	12
2.9 Java	13
3. Arquitetura para Web proposta pela Oracle	16
3.1 Internet Application Server (IAS)	16
3.2 Banco de dados Oracle 9i	16
3.3 Oracle Designer	18
3.4 Forms Developer	18
3.4.1 Forms Services.....	19
3.4.1.1 Componentes do Forms Services.....	20
3.4.2 Métodos de conexão usados pelo Forms Services	21
3.4.3 Configuração do Browser	22
3.4.4 Execução das aplicações Forms	22
3.4.5 Cuidados a serem tomados pelo desenvolvedor Web	22
3.4.6 Sugestões da Oracle para melhorar o desempenho do aplicativo	23
3.5 Reports Developer	25
3.5.1 Reports Services	25
3.6 Oracle9i Development Suite	26
3.6.1 Componentes do <i>Oracle9i Development Suite</i>	27
3.6.2 Jdeveloper	27
4. Conclusão	29
Referências Bibliográficas	30
Apêndice A – Telas Principais das ferramentas da Oracle	31
Figura A1 – Tela Principal do Oracle Designer 6.0	31
Figura A2 – Tela Principal do Forms Builder	32
Figura A3 – Tela Principal do Report Builder	35
Apêndice B – Exemplos de Implementação de Recursos na Base de Dados	37
Figura B1 – Exemplo de uma <i>PL/SQL</i>	37

Figura B2 – Exemplo de uma <i>procedure</i>	38
Figura B3 – Exemplo de uma função	39
Figura B4 – Exemplo de uma <i>package</i>	40
Figura B5 – Exemplo de <i>Database Trigger</i>	41

Lista de Figuras

Figura 2.1 – Arquitetura Web simplificada	4
Figura 2.2 – Arquitetura Cliente/Servidor	5
Figura 2.3 – Esquema de um ambiente de três camadas	6
Figura 2.4 - Estrutura de banco de dados relacional	8
Figura 2.5 - Estrutura do bloco PL/SQL	10
Figura 2.6 - Ambiente <i>Java</i>	15
Figura 3.1 – Arquitetura Web com o servidor de aplicações e componentes do Middleware .	16
Figura 3.2 – Arquitetura Web mostrando os componentes do Forms Services	21

Lista de Abreviaturas

Sigla	Significado
DBA	Data Base Administrador
FTP	File Transfer Protocol
CGI	Graphical User Interface
HTML	HyperText Markup Language
HTTP	HyperText Transfer Protocol
OLE	Object Linking and Embedding
PDF	Adobe Portable Document Format
RAD	Rapid Application Development
SQL	Structured Query Language
SSL	Secure Sockets Layer
TCP/IP	Transmission Control Protocol/Internet Protocol
URL	Uniform Resource Locator
XML	Extensible Markup Language
WAIS	Wide Area Information Server

1. Introdução

Em um mundo globalizado como o nosso a *Internet* viabiliza a comunicação entre pessoas e empresas, tornando-se um meio rápido para disseminar informações. Com a criação de aplicações para *e-business*¹ cria-se redes colaborativas entre as organizações, que podem ser acessadas a qualquer momento e de qualquer lugar. Transações entre empresas e clientes através de sistemas na *Web* são feitas em poucos minutos e até segundos. Com isto as empresas buscam ganhar mais agilidade, baixar custos operacionais e assim tornarem-se mais competitivas. A rapidez com que as informações são atualizadas representa uma arma poderosa para as organizações. Para isto, o mercado disponibiliza várias ferramentas e banco de dados para o desenvolvimento de sistemas para *Web*. É necessário que os profissionais da área de informática conheçam as diversas ferramentas disponíveis no mercado, a fim de poderem utilizar a que mais se adapta a realidade da sua empresa tirando proveito desta grande rede de alcance mundial, a *Internet*², evitando desperdício de tempo e dinheiro. Neste trabalho será apresentado o ambiente integrado de desenvolvimento de sistemas para *Web* da *Oracle Corporation*³.

1.1 Objetivos

Neste trabalho serão disponibilizadas informações detalhadas sobre a tecnologia oferecida pela *Oracle* para desenvolvimento de sistemas para *Web*. Será feita uma breve apresentação da *World Wide Web*. Serão mostradas algumas das empresas que estão utilizando a tecnologia da *Oracle* e em quais sistemas. Será descrito o conjunto de programas *Forms Services e Reports Services*, que fornecem uma infraestrutura para disponibilização de aplicações na *Web*, sem que os desenvolvedores de aplicativos tenham grandes preocupações com os protocolos de comunicação e serviços de rede. Será abordado o conceito utilizado pela *Oracle* sobre o ambiente de três camadas, sendo um cliente, um servidor de aplicação e um servidor de banco de dados. Será mostrado o principal componente da *Oracle Internet Architecture* utilizado pelo banco de dados *Oracle 9i* e pelo servidor de aplicações *Web*.

¹ *e-business*: Negócios praticados através da internet

² *Internet*: rede mundial de computadores.

³ *Oracle Corporation*: Empresa com sede em *Redwood Shores*, na Califórnia. Com faturamento anual superior a 10,8 bilhões de dólares. Oferece produtos de bancos de dados, ferramentas e aplicativos, bem como serviços

Também serão vistas as ferramentas de desenvolvimento de sistemas da Oracle: *Forms Server*, *Reports Server*, *Oracle Designer*, *Oracle9i Development Suite* e *Jdeveloper*. Não faz parte do foco deste trabalho ensinar como utilizar as ferramentas, e sim descrever as vantagens, restrições e peculiaridades no desenvolvimento de sistemas para a *Web* utilizando estas ferramentas. Serão descritas as restrições na criação de aplicações *Forms* e *Reports* para serem executadas na *Web*. A abordagem na utilização do *Forms* e *Reports* no desenvolvimento de aplicações será mais minuciosa, ajudando os desenvolvedores que já utilizam esta ferramenta a conhecer as peculiaridades na utilização destas nas aplicações para *Web*. Será apresentado também o que a *Oracle* disponibiliza sobre a segurança destas aplicações. Serão mostrados os requisitos necessários para o desenvolvimento e implementação de uma aplicação *Forms* em um ambiente *Internet*. Com este trabalho, pretende-se disponibilizar aos profissionais um material detalhado, o qual poderão analisar de forma a conhecer a arquitetura *Web* proposta pela *Oracle*, ajudando com isto, aumentar os conhecimentos e também oferecer subsídios para a escolha do ambiente de desenvolvimento de sistemas para *Web*.

1.2 Organização

O restante do trabalho está organizado conforme descrito a seguir:

O segundo capítulo traz uma apresentação da *World Wide Web*, uma abordagem da arquitetura utilizada pela *Oracle*, onde é visto o conceito da arquitetura multi-camadas. Será mostrado o esquema de um ambiente de três camadas e as responsabilidades de cada uma e a arquitetura cliente/servidor. É também feita uma breve abordagem de como o *Web Application Server* trata a segurança. Neste capítulo são tratados tópicos importantes para a compreensão da arquitetura da Oracle, tais como, bancos de dados, Interface Gráfica do Usuário, *SQL*, *PL/SQL* e *Java*. No terceiro capítulo serão descritos o banco de dados *Oracle 9i*, o Servidor de Aplicação e as ferramentas para desenvolvimento de sistemas para *Web* oferecidas pela *Oracle*. Por fim, no Capítulo 4 contém a conclusão do trabalho.

2. Revisão Bibliográfica

2.1 World Wide Web

A *World Wide Web*, ou simplesmente *Web*, é um serviço oferecido pela *Internet*. A *Web* é um recurso globalmente distribuído residindo sob a rede mundial de computadores, a *Internet*. A *Internet* foi fundada pelo Departamento de Defesa dos Estados Unidos da América em 1969, originalmente chamada de *Arpanet* (Gabriel,2001).

A *Internet* é uma rede onde milhares de computadores se comunicam e compartilham funções de administração de rede de acordo com um conjunto de regras pré-estabelecidas.

Uma característica da *Web* é que ela foi projetada para funcionar no “topo” da *Internet*, que é composta por uma grande variedade de computadores e redes que interagem entre si, que permite acesso a documentos ligados e espalhados por milhares de máquinas na *Internet*. Sendo utilizada para transmissão e disseminação de informações, acesso a dados e comunicações (Lima,1997).

Uma Segunda característica é que a *Web* foi projetada para encapsular vários protocolos *Internet* incluindo, entre outros, FTP⁴, GOPHER, WAIS⁵, NNTP, e TELNET. Assim é eliminada a necessidade de se usar um programa separado para cada serviço *Internet* diferente, unificando-os num único serviço, no caso a *Web* (Lima,1997).

E uma terceira característica importante da *Web* é que ela é baseada nos princípios apresentados a seguir (Lima,1997):

1. **Transferir a informação:** Para o transporte de informações entre o servidor e o cliente *Web* é utilizado o protocolo de comunicação HTTP⁶ como demonstrado na Figura 2.1. Este protocolo é especializado na transmissão de documentos *Web* na *Internet*.
2. **Descrever a formatação da informação:** Para apresentação e formatação de documentos na *Web* é utilizada a linguagem padrão HTML⁷, que permite que a estrutura dos documentos *Web*, bem como os vínculos (*links*) e recursos da *Internet*, sejam incorporados diretamente em formatos textos.

⁴ FTP : File Transfer Protocol

⁵ WAIS : Wide Area Information Server

⁶ HTTP : *Hypertext Transport Protocol*. É o protocolo que define como é que dois programas/servidores devem interagir, de maneira a transferirem entre si comandos ou informações relativos ao WWW.

⁷ HTML : *Hyper Text Markup Language*.

3. **Localizar a informação:** Para identificação e localização de documentos *Web* distribuídos pela *Internet* é utilizado o formato para endereçamento de documentos *Web* denominado URL⁸. A sintaxe de uma URL é *<Protocolo>://<Host>/<Path>/<Doc>[<#Location>]*. *Protocolo* indica o tipo de recurso *Internet* que deve ser usado para a conexão com os servidores, *Host* é o nome da máquina, *Path* é uma lista de diretórios separada por barras, *Doc* é o nome do documento ou programa a ser executado pelo servidor *Web* e *Location* é uma marca textual opcional de posição de documento. Ex: <http://www.simplex.com.br/ids.html>.

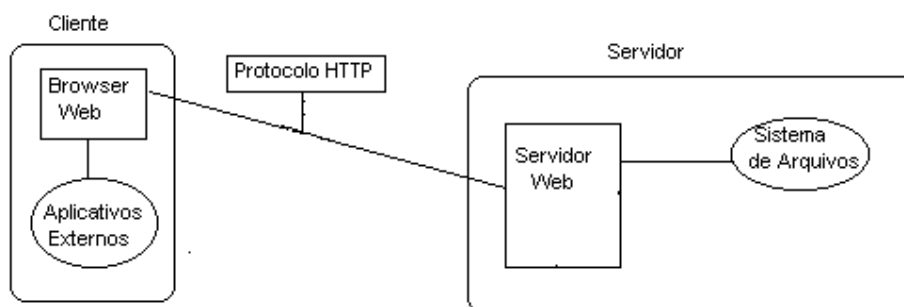


Figura 2.1 – Arquitetura Web simplificada

2.2 Arquitetura Cliente/Servidor

Esta arquitetura se baseia em um servidor de banco de dados e o cliente, que representa as aplicações que acessam o banco de dados e disponibiliza a interface para o usuário.

Em um ambiente Cliente/Servidor o *Forms Runtime*⁹ é executado na máquina do cliente e a atualização no banco de dados é efetuada no servidor. Toda a interface com o usuário e lógicas da aplicação podem ser executadas na máquina do cliente, portanto, neste caso estas máquinas devem ter capacidade de processamento e memória suficiente para a execução das aplicações localmente (Fernandes, 2002).

⁸ URL : *Uniform Resource Locator*

⁹ *Forms Runtime* : componente que executa os aplicativos

Na arquitetura Cliente/Servidor as máquinas dos clientes podem estar conectadas por redes locais e compartilham o processamento das aplicações com os servidores de rede. As redes locais podem estar conectadas a outras redes. Este ambiente Cliente/Servidor de várias redes locais interconectadas pode substituir uma grande rede baseada em *mainframe*¹⁰ com muitos terminais de usuários finais (Gabriel, 2001). A arquitetura cliente/servidor é demonstrada na Figura 2.2 , de um lado fica o cliente e no outro o servidor de banco de dados atendendo as solicitações do cliente.

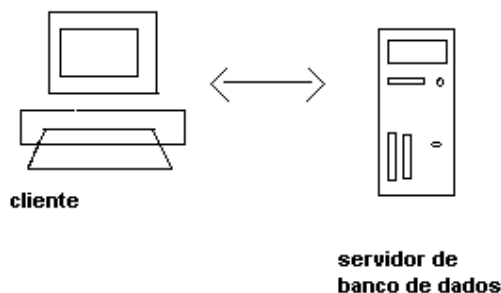


Figura 2.2 – Arquitetura Cliente/Servidor

2.3 Arquitetura Multi-camadas

A arquitetura multi-camadas é aquela em que o sistema de informação é dividido em unidades lógicas que compartilham dados e se comunicam (Gabriel,2001).

No caso do ambiente *Web* proposto pela *Oracle* é utilizada a arquitetura de três camadas que é constituído de um cliente e dois servidores como apresentado na Figura 2.3. O cliente possui um *browser* , e é onde acontece o processamento da interface, exibindo e solicitando documentos. O banco de dados esta em um servidor, idêntico ao modelo tradicional (cliente/servidor), e as aplicações ficam no servidor de aplicações, onde esta o executável do *Forms*¹¹ e os serviços de *runtime*, podendo ser constituído de uma máquina ou um pool¹² de máquinas.

¹⁰ *mainframe* : sistema de computador de grande porte, normalmente com uma unidade central de processamento independente, distinto dos sistemas de micro e minicomputadores.

¹¹ *Forms* : Aplicativos

¹² *pool* : coleção

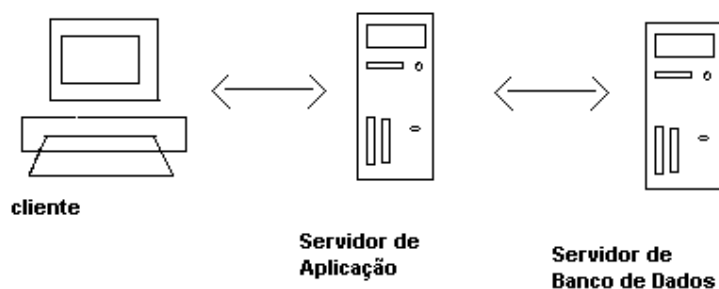


Figura 2.3 – Esquema de um ambiente de três camadas

A arquitetura de três camadas se divide em (Gabriel,2001):

- cliente: Camada de apresentação. É responsável pela lógica da apresentação e controle da interação entre o usuário e o computador.
- servidor de aplicação: É responsável pela lógica da aplicação, pelas decisões, cálculos e operações que a aplicação deve realizar.
- servidor de banco de dados: Camada de acesso a dados localizada no servidor de banco de dados. É responsável pelos dados, metadados e regras de integridade.

A arquitetura em camadas possui as características necessárias para atender às necessidades do mercado: flexibilidade, escalabilidade, segurança e integração (Gabriel,2001). As diversas camadas podem sofrer ajustes e evolução sem afetar toda a solução. Esta arquitetura permite que a complexidade e processamento das informações fiquem a cargo do servidor que processa e formata as requisições do cliente, ficando para este apenas a apresentação dos dados.

2.3.1 Algumas das vantagens da utilização da arquitetura de 3 camadas:

- . A máquina do cliente não necessita de grande capacidade de processamento, pois a execução dos requisitos dos clientes será processada no servidor de aplicação.
- . Maior velocidade de acesso ao Banco de Dados, pois o servidor de aplicações é geralmente máquinas mais potentes.
- . Oferece a flexibilidade de não sobrecarregar o cliente.

. Esta arquitetura resolve a dificuldade de implementar a lógica do sistema de informação no *browser*, a deficiência das linguagens *scripts*¹³ para *browsers* que oferecem suporte à manipulação de arquivos ou banco de dados; e a dificuldade de transmissão devido à carga da rede (Gabriel, 2001).

2.4 Segurança

A *Web* apresenta grandes desafios com relação a segurança. A transmissão de dados via protocolo padrão HTTP não é segura, podendo ser interceptadas na rede. Em muitas aplicações *Web* baseadas em acesso a banco de dados, é requerido que as entidades comunicantes, clientes e servidores *Web*, se autenticuem e certifiquem mutuamente de forma a garantir a privacidade das informações. Vários requisitos de segurança devem ser satisfeitos: privacidade através do uso de mecanismos de criptografia, autenticação de servidores e clientes *Web* e manutenção da integridade de transações, de forma a se determinar se os dados recebidos foram corrompidos ou falseados.

Segundo a *Oracle* o *Web Application Server* suporta segurança no cliente, em cada nível dentro da arquitetura do servidor *Web* e através do firewall¹⁴, até um banco de dados *Oracle*. Permitindo não apenas proteção do tipo identificação/senha, mas também para segurança ainda maior de dados críticos, o *Web Application Server* suporta o padrão de mercado SSL3¹⁵ para criptografia, para garantir comunicações e transações seguras.

O conjunto de programas e serviços que compõem a infraestrutura *Forms Services* para disponibilização de aplicações na *Web*, utiliza regras de segurança SSL com criptografia de mensagens, impedindo que estas sejam modificadas e autenticação e verificação do servidor que quer acessar o sistema.

2.5 Banco de Dados

A definição de banco de dados é de um conjunto de informações armazenadas de forma organizada. Um sistema de banco de dados envolve os próprios dados, o hardware em que os dados residem, o software que controla o armazenamento e a recuperação de dados,

¹³ *Scripts* : roteiros

¹⁴ *Firewall* : equipamento que serve como *gateway* de segurança entre uma intranet e a internet. Eles controlam a fonte e o destino do tráfego da rede e o filtram.

¹⁵ SSL3 : protocolo de segurança – SSL – *Secure Sockets Layer*

chamado de sistema de gerenciamento de banco de dados (RDMS) e os próprios usuários. O software DBMS gerencia os pedidos dos usuários para acesso às informações, controla o armazenamento, recuperação e a modificação dos dados. O banco de dados Oracle 9i tem como gerenciador o RDBMS que inclui um gerenciador de banco de dados e diversas ferramentas para gerenciar a armazenagem e definição de dados, controlar e restringir o acesso e concorrência aos dados, permitir “back-up” e recuperação de dados e interpretar os comando SQL. O sistema de gerenciamento do *Oracle 9i* é objeto-relacional.

Um banco de dados relacional é uma coleção de tabelas bi-dimensionais, a recuperação de dados é feita através de operações relacionais sobre estas tabelas. Na Figura 2.4 tem-se um exemplo de tabela em uma estrutura de banco de dados relacional, a tabela artigos possui uma chave estrangeira *cod_fornecedor* que é chave primária na tabela fornecedores. O formato linha/coluna de uma tabela é a maneira usual de visualizar os dados.

Tabela: artigos

codigo	descrição	cod_fornecedor	----- coluna
1	geladeira	101	----- linha/registro
2	fogão	110	
3	televisor	105	

chave primária

Tabela: fornecedores

Cod_fornecedor	nome
101	Consul S/A
102	Brastemp S/A
105	Phillips S/A
110	Daco S/A

Figura 2.4 - Estrutura de banco de dados relacional.

Esta representação lógica dos dados permite considerar relacionamentos entre os dados sem se envolver com a implementação física das estruturas de dados (Deitel, 2001). A partir da Versão 8 do banco de dados da *Oracle* é possível armazenar e obter dados de

objetos. Numa base de dados relacional de objetos é possível utilizar o *SQL* e o *PL/SQL*¹⁶ para manipular dados relacionais e objetos (Dorsey, 1999).

Algumas das vantagens dos sistemas de banco de dados são (Deitel, 2001):

- . A diminuição da redundância de dados é reduzida, pois cada aplicativo não possui seu próprio arquivo de dados, estes estão armazenados no mesmo banco de dados.
- . Como todos os aplicativos acessam o mesmo banco de dados o compartilhamento destes dados é uma grande vantagem.
- . É mais fácil manter a integridade das informações armazenadas no banco de dados.
- . Os aplicativos não precisam se preocupar com a maneira que os dados estão armazenado fisicamente, ficando a critério do sistema de gerenciamento do banco de dados esta preocupação.

Um banco de dados pode ser distribuído, isto é um banco de dados que é espalhado pelos sistemas de computador de uma rede, geralmente cada item de dados é armazenado aonde eles são mais freqüentemente utilizados e acessíveis para outros usuários da rede (Deitel, 2001)

2.6 GUI – Interface Gráfica do Usuário

Para a construção do *front-end*, isto é, a interface que permite aos usuários acessar os dados e executarem os aplicativos são usadas interfaces gráficas (GUI) (Gabriel, 2001). A *Oracle* disponibiliza a ferramenta *Forms Developer* onde são criadas as telas de manipulação de dados com uma interface gráfica do usuário dirigida para mouse, que possibilitam aos usuários finais acessarem informações armazenadas em um banco de dados.

2.7 SQL – Structure Query Language

O *SQL* é uma linguagem de programação não procedural utilizada pelos bancos de dados relacionais.

O *SQL* contém vários comandos para execução de tarefas tais como:

- . Consulta, inserção, atualização, remoção de linhas na tabela.
- . Criação, modificação e remoção de objetos do banco de dados.

¹⁶ *PL/SQL* : São blocos armazenados na base de dados para repetidas execuções , que podem ser invocados pelo *SQL*, por outras ferramentas ou por outras procedures e funções armazenadas na base de dados.

. Controle de acesso ao banco de dados e seus objetos.

O Gerenciador de Banco de dados da *Oracle* utiliza o *SQL* como linguagem de consulta, permitindo ao usuário acessar os dados. A linguagem *SQL* não necessita que seja especificado o método de acesso aos dados, isto é feito pelo RDBMS que utiliza um otimizador para determinar a maneira mais rápida de recuperar os dados. A interface entre o usuário e o banco de dados da *Oracle* é feita através do *SQL*PLUS* , no qual pode-se entrar e executar comandos *SQL*.

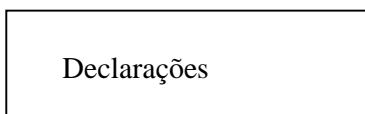
2.8 PL/SQL – Procedural Language for SQL

O *PL/SQL* é uma linguagem procedural que pode ser usado nas ferramentas *Oracle*. Linguagem de extensão ao *SQL* permite implementar recursos ativos na base de dados tais como: *Procedures*, Funções, *Packages* e *Database Triggers* (*Triggers* na base de dados). Com o *PL/SQL* podem ser montadas rotinas de tarefas tais como: atualizar tabelas, criar cursores que recebem um registro ou parte dele de uma tabela e trabalhar o resultado da *query* executada, usar loops para executar repetidamente enquanto uma condição não é satisfeita.

O *PL/SQL* tem uma parte inteiramente dedicada a tratamento de erros ocorridos, que podem ser os vários já pré-definidos ou algum que o usuário queira especificar, chamados de *exception*. Quando um erro ocorre, uma *exception* que testa esse erro é acionada, e então, se o teste for verdadeiro e o erro realmente ocorreu, uma estrutura de falha *raise* é acionada e todo o fluxo do programa passa para o *raise*. Se nenhum erro foi previsto, isto é, não foi definida uma *exception*, mas um erro ocorreu o programa continua a execução da lógica sem o tratamento do erro.

Na Figura 2.5. é mostrada a estrutura do bloco *PL/SQL* , na parte do *DECLARE* é feita as declarações das variáveis, na parte entre o *BEGIN* e *END* esta o corpo de comandos da *PL/SQL* e na parte da *EXCEPTION* é dedicada ao tratamento de erros ocorridos, que podem ser os vários já pré-definidos ou algum que o usuário queira especificar.

DECLARE



BEGIN

Estruturas executáveis e
outros blocos PL/SQL

EXCEPTION

Tratamentos de Exceções

END;

Figura 2.5 - Estrutura do bloco PL/SQL

2.8.1 Procedures e Funções

Procedures e Funções são blocos PL/SQL armazenados na base de dados para repetidas execuções, que podem ser invocados pelo SQL, por outras ferramentas ou por outras *procedures* e funções armazenadas na base de dados. A diferença entre *procedures* e funções é que a *procedure* pode conter uma lista de argumentos, e pode retornar um ou mais valores e a função pode conter uma lista de argumentos, e deve retornar apenas um valor. Na Figura B1 tem um exemplo de *procedure* e na Figura B2 no apêndice B tem um exemplo de função.

Na criação de uma *procedure* a transferência de valores, isto é na passagem de parâmetros os tipos de argumentos podem ser:

IN - Passa um valor do ambiente chamador p/ *procedure*

OUT - Retorna um valor da *procedure* para o ambiente chamador.

IN OUT - Passa um valor do ambiente chamador para a *procedure*, e retorna um outro valor da *procedure* para o ambiente chamador.

Vantagens na utilização de *procedures* e funções:

- . melhora a segurança, pois controla os acessos indiretos aos objetos do banco de usuários sem permissão.

- . controle na integridade dos dados, pois assegura que ações relacionadas sejam executadas conjuntamente.

- . melhora o desempenho, pois reduz o número de chamadas ao banco e permite compartilhar execuções SQL por vários usuários.

- . conserva a memória , pois ao invés de várias cópias espalhadas dentro dos aplicativos, é armazenada uma única cópia no banco.

- . otimiza as manutenções futuras, pois é feita uma única modificação que afetará várias aplicações.

2.8.2 Package

Pode-se definir *package* como sendo um grupo de identificadores de programação e rotinas armazenadas juntas como um pacote. Mais utilizado para agrupar procedures e funções.

Uma *Package* é dividida em duas partes:

Especificação – declara-se as construções públicas. Disponível para procedures e funções mesmo fora da package.

Corpo - declara e define as construções utilizadas no corpo da package. Define o corpo das construções públicas.

Vantagens na utilização de packages:

- . melhora a organização de procedures e funções
- . melhora o gerenciamento de procedure e funções armazenadas, pois ao alterar o corpo da package a especificação da package não é alterada.
- . melhora a segurança de procedures e funções armazenadas, pois permite acessar um pacote inteiro por vez e oculta o código fonte do usuário.
- . melhora a performance, pois quando uma package é chamada esta carrega para a memória o pacote inteiro, reduzindo acessos em disco para posteriores chamadas.

Na Figura B3 do apêndice B é exemplificado uma *package*.

2.8.3 Database Triggers

Database Triggers é um bloco *PL/SQL* que será executado sempre que uma tabela sofra algum tipo de manipulação de dados.

Existem dois tipos de *Database Triggers*:

. ***Database Triggers de comando*** - São comandos que manipulam um grupo de dados dentro de uma tabela e executam uma única vez, como por exemplo, *delete*¹⁷ para um grupo de linhas.

Geralmente estas *triggers* de comando são usadas para:

- . prevenir operações com dados inválidos.
- . auditoria de operações de dados.
- . controlar a segurança de operações de dados.
- . inicializar e resetar variáveis globais.

No Anexo G é exemplificada uma *database triggers* de comando.

. ***Database Triggers de linha*** – São comandos que manipulam linhas de uma tabela e que podem ser executadas uma ou mais vezes, como por exemplo, *insert*¹⁸ e *update*¹⁹ em linhas específicas.

Dentro de uma *trigger* de linha (*For each row*) para se referenciar ao valor de uma coluna antes dos dados serem alterados deve-se utilizar o qualificador *old*, e para referenciar seu valor depois dos dados serem alterados deve-se utilizar o qualificador *new*.

Os qualificadores *old* e *new* são utilizados dentro de *triggers* de linha *before* (antes) e *after* (depois) . Estes qualificadores não podem ser usados dentro de *triggers* de comandos. Estes qualificadores deverão ser precedidos de dois pontos como demonstrado na Figura B4 do apêndice B .

Geralmente estas *triggers* de comando são usadas para:

- . prevenir dados inválidos dentro de linhas.
- . auditoria de linhas e valores afetados por operações de dados.
- . controle dos dados e integridade referencial.
- . usar variáveis globais e flags para controlar operações de dados.
- . replicar tabelas.
- . calcular valores derivados transparentemente.

Na Figura B5 do apêndice B é exemplificada uma *database triggers* de linha.

2.9.1 – Java

¹⁷ *delete* : comando de deleção de informações na base de dados

¹⁸ *insert* : comando de inserção de informações na base de dados

¹⁹ *update* : comando de alteração de informações na base de dados.

Java é uma linguagem de programação orientada a objetos. Foi criada e lançada como um meio de adicionar mais conteúdo dinâmico tais como: áudios, vídeos, animações e interatividade.

A Linguagem *Java* também foi projetada para atender às necessidades do desenvolvimento de aplicações em um ambiente distribuído e heterogêneo. Uma das características principais da Linguagem *Java* é sua simplicidade. Java é utilizado na criação de páginas da Web com conteúdo interativo e dinâmico, para desenvolver aplicativos corporativos de larga escala, para aprimorar a funcionalidade de servidor *Web* e fornecer aplicativos para dispositivos como telefones celulares, pagers e PDAs (máquinas utilizadas em lojas) (Deitel,2001). A linguagem é acompanhada de um grande número de bibliotecas de classes já testadas e pode ser estendida via herança. As bibliotecas de classes são também conhecidas como *Java APIs*²⁰, interfaces de programas aplicativos.

O Código *Java* pode ser executado em qualquer máquina que possua o interpretador *Java*.

O Java permite aos programadores escrever código que utiliza consultas de *SQL* para acessar informações em sistemas de banco de dados relacional.

Segundo Deitel (Deitel, 2001), os Programas *Java* normalmente passam por cinco fases para serem executadas:

. Edição – Edita o arquivo com um programa editor, ao digitar o programa o programador já faz as correções necessárias. Os arquivos com o programa Java terminam com a extensão java.

. Compilação – Através do comando javac o programa é compilado. O compilador *Java* traduz o programa para *bytecodes*²¹.

. Carga – O carregador de classe transfere o arquivo .class contendo o programa para a memória. Existem dois tipos de programas para os quais o carregador de classe carrega arquivos .class, aplicativos e *applets*. Aplicativo é um programa que normalmente é armazenado e executado a partir do computador local do usuário, para se executar um aplicativo é utilizado o interpretador *Java* através do comando java. O *applets* é um programa carregado no navegador a partir de um computador remoto, são executados no navegador e

²⁰ *APIs – Applications Programming Interfaces*

descartados quando se completa a execução ou também podem ser executados a partir da linha de comando utilizando o comando `appletviewer` (navegador mínimo – interpreta apenas *applets*). O carregador de classe também é executado quando um *applet Java* é carregado em um navegador da *Web* como: *Netscape Communicator*, o *Internet Explorer da Microsoft* ou o *HotJava da Sun*. Os navegadores são utilizados para visualizar documento na *Web* chamados documento HTML²².

. Verificação – A verificação é feita antes da execução pelo interpretador *Java* os *bytecodes* em um *applet*.

. Execução - Através do controle da CPU o computador interpreta o programa, um *bytecode* por vez, e executa a ação do programa.

Na Figura 2.6 mostra as fases que um programa java passa ao ser executado.

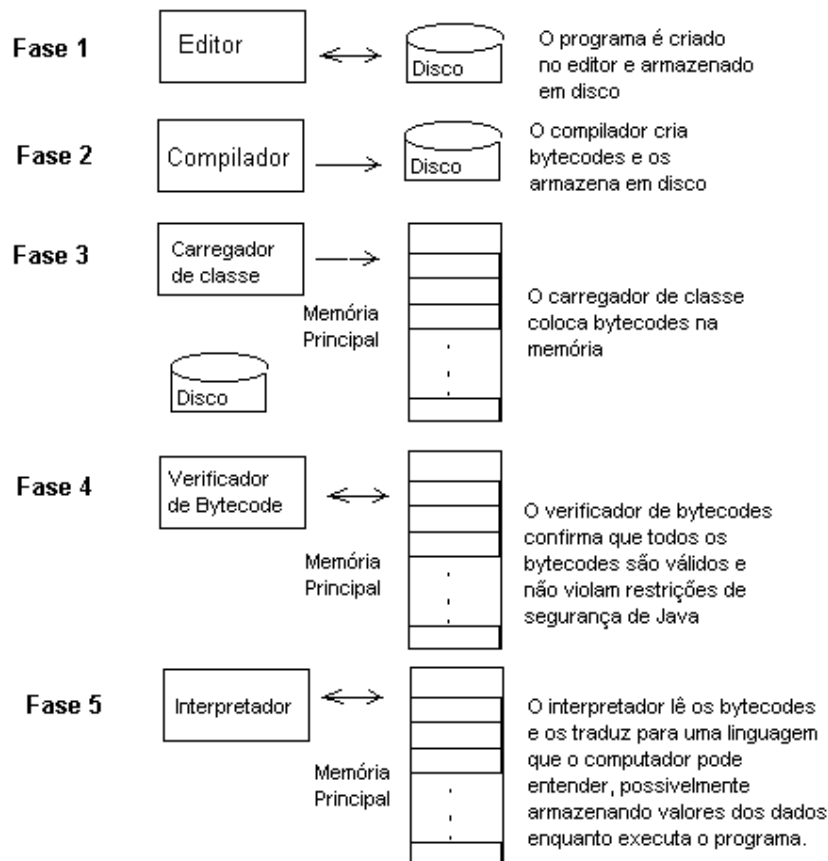


Figura 2.6 - Ambiente Java (Deitel,2001)

²¹ *bytecodes* : termo utilizado em *Java* para designar o código intermediário, isto é, o código no qual são representadas as instruções do programa que serão posteriormente interpretadas e executadas pelo interpretador *Java*.

²² HTML – Hypertext Markup Language

3. Arquitetura para Web proposta pela Oracle

3.1 Internet Application Server (IAS)

O *Internet Application Server* é um servidor que integra um conjunto de componentes do *middleware*²³ como demonstrado na Figura 3.1, incluindo *web server*, *middleware* de acesso ao banco de dados, monitores de processamento de informação, entre outros, para a comunicação entre o *browser* do cliente e o servidor *Web* é usado o protocolo HTTP. Suporta os padrões das linguagens para *Internet* (*Java*²⁴, *Perl* etc.) e garante alta performance através de balanceamento de carga. Oferece escalabilidade, confiabilidade e facilidade de gerenciamento (Simplex,2003).

O *Oracle Internet Application Server* ajuda eliminar camadas extras de lógica, e os custos inerentes de desenvolvimento de controles já existentes no IAS, enquanto fornece uma base sólida para o crescimento e serviços de alta qualidade para os usuários (Simplex,2003).

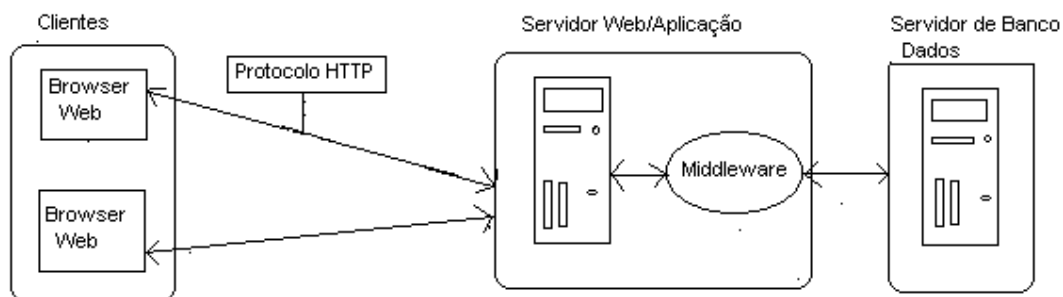


Figura 3.1 – Arquitetura Web com o servidor de aplicações e componentes do middleware

3.2 Banco de dados Oracle 9i

O Banco de dados *Oracle 9i*, com seu sistema de gerenciamento de *Banco de Dados Objeto-Relacional* está apto a gerenciar um grande volume de dados em um ambiente multi-usuário, armazenando e recuperando os dados solicitados ao mesmo tempo por vários usuários (Gabriel,2001). O Banco de dados *Oracle 9i* possui recursos de integridade

²³ *middleware* : software intermediário em uma arquitetura, que faz a integração de tecnologias.

²⁴ *Java* : linguagem muito utilizada no desenvolvimento de aplicações para a Internet. Linguagem simples e orientada a Objetos.

referencial, permite o uso de partições para melhorar a eficiência, de replicação e administração de bancos de dados distribuídos. Os sistemas distribuídos fornecem controle e economia de processamento local com as vantagens de acesso de informações geograficamente espalhadas. Pode ser executado em vários sistemas operacionais.

Para usufruir os recursos do Banco de dados *Oracle 9i* é fundamental a presença constante de um DBA²⁵. Um administrador de banco de dados tem conhecimento técnico e detalhado sobre a funcionalidade do banco de dados podendo com isto, obter o melhor desempenho e utilizar os recursos fornecidos pelo banco de dados *Oracle 9i*. Para garantir a eficiência e a rapidez é necessária a realização da instalação seguida da configuração de maneira correta e o DBA é a pessoa mais indicada para executar esta tarefa.

Principal componente da *Oracle Internet Architecture*, segundo a *Oracle* o banco de dados *Oracle 9i* foi desenhado para suportar vários tipos de dados, possibilitando um sistema de alto desempenho e com excelente relação custo-benefício para aplicações de negócios, oferece proteção à acessos não autorizados, através do controle de *grant*²⁶ e soluções eficientes para recuperação de dados ocasionados por falhas. É um banco de dados pronto para a *Internet*.

O *Oracle 9i* é uma plataforma de modelo computacional baseado na *Internet* que permite que qualquer tipo de dado seja gerenciado a partir de um servidor centralizado e acessado de qualquer cliente, em qualquer rede. Projetado para usufruir a eficiência de custos da rede mundial. O *Oracle 9i* está disponível em três edições: *Enterprise*, *Standard* e *Personal* (Oracle,2003).

O Banco de dados *Oracle 9i* apresenta incrementos, sobre as versões anteriores, principalmente alta disponibilidade, melhorias para manipulação de *data warehouses*²⁷ e suporte a linguagem *Java*. Funcionalidades adicionais e melhorias de performance também estão entre os seus destaques.

O Banco de Dados *Oracle9i* utiliza a linguagem de consulta *SQL* que foi apresentada na Seção 2.7. O *SQL* fornece duas sublinguagens de banco de dados (*DSL*²⁸) voltadas para as

²⁵ DBA : Administrador de Banco de Dados

²⁶ *grant* : através deste comando concede privilégios de acesso a sistemas e roles, bem como privilégios de cesso a objetos para usuários. Podendo restringir a consulta, alteração ou inserção em uma determinada tabela ou coluna no banco de dados.

²⁷ *data warehouse* : conjunto de dados extraídos do banco de dados, selecionados, editados e padronizados para recuperação e análise, ajudando na tomada de decisões gerenciais.

²⁸ *DSL* : *database sublanguage*

especialidades dos objetos e operações do banco de dados; DDL²⁹ é uma linguagem de definição de dados, fornece recursos para definir os objetos do banco de dados, e DML³⁰ é uma linguagem de manipulação de dados, fornece recursos para especificar o processamento a ser realizado sobre objetos de banco de dados (Deitel,2001).

Através do *PL/SQL*³¹, linguagem procedural, pode-se implementar recursos ativos, como *database triggers*³² e *stored procedures*³³.

3.3 Oracle Designer

Oracle Designer oferece um ambiente intuitivo de modelagem que permite ao desenvolvedor compreender claramente e corresponder aos requisitos comerciais. Fornece suporte para várias abordagens de modelagem visual, incluindo diagrama de ER, diagrama de funções, análise e *design* de objetos. Permite geração automática do esquema da base de dados e possui também recursos de engenharia reversa (Simplex,2003). O *Oracle Designer* oferece ferramentas para modelar seu aplicativo e seu banco de dados, e para gerar aplicativos cliente/servidor e baseados na *Web*. Com seu ambiente de trabalho integrado, o *Oracle Designer* oferece um grande aumento de produtividade para os desenvolvedores, pois gera automaticamente aplicativos baseados nas informações contidas no seu repositório. E também uma base centralizada de informações sobre todos os sistemas da empresa. Através de seus relatórios é possível evitar a redundância e duplicação de informações dentro dos sistemas integrados da empresa. Na Figura A1 do apêndice A é apresentado a tela principal do *Oracle Designer*.

3.4 Forms Developer

O *Forms Developer* corresponde a um dos ambientes de produtividade da *Oracle - RAD*³⁴ capaz de construir com rapidez aplicações a partir das definições do banco de dados, podendo fazer manutenções na base de dados (Oracle,2003).

²⁹ DDL: *data definition language*

³⁰ DML : *data manipulation language*

³¹ *PL/SQL* : São blocos armazenados na base de dados para repetidas execuções , que podem ser invocados pelo SQL, por outras ferramentas ou por outras procedures e funções armazenadas na base de dados.

³² *Database triggers* : Um bloco PL/SQL que será executado sempre que uma tabela sofra algum tipo de manipulação de dados.

As aplicações criadas são disponibilizadas em ambiente *Web*, integradas com a solução *Oracle*, seja banco de dados, *Oracle Designer* ou Servidor *Web* de Aplicação.

O *Forms Builder 6i* é a principal ferramenta deste ambiente. Na execução do *forms* temos a opção de simularmos a execução de uma aplicação em ambiente *Web*. Na Figura A2 do apêndice A é apresentado a tela principal do *Forms Developer* e descrito o funcionamento da estrutura dos nós do navegador de objetos.

Do pacote *Forms Developer* fazem parte também as seguintes ferramentas de apoio:

- . **Procedure Builder** – Auxilia na criação de programas PL/SQL, tanto no ambiente cliente com bibliotecas de programas compartilhadas por aplicações *forms*, quanto no ambiente do servidor com *procedures*, *functions*, *packages* e *database triggers*. Permite depuração passo a passo.

- . **Graphics Builder** - Permite visualizar os dados do banco de dados de forma gráfica.

- . **Project Builder** – Integra e organiza as diversas aplicações em um projeto com uma hierarquia manipulável.

- . **Query Builder** – Auxilia na construção de consulta ao banco de dados.

- . **Schema Builder** – Auxilia na criação, cópia, modificação e remoção de objetos do banco de dados.

- . **Translation Builder** – Ferramenta de suporte e gerenciamento no processo de conversão de aplicações Oracle para uma das linguagens suportadas.

O *Form Builder* é capaz de construir três tipos diferentes de módulos³⁵, isto é, tipos de aplicativos: módulo Form³⁶, módulo Menu³⁷, módulo PL/SQL Library³⁸ e também arquivos de Object Library³⁹ (Fernandes,2002).

3.4.1 *Forms Services*

³³ *Stored Procedure* : São blocos PL/SQL armazenados na base de dados para repetidas execuções , que podem ser invocados pelo SQL, por outras ferramentas ou por outras *procedures* e funções armazenadas na base de dados.

³⁴ RAD : Rapid Application Development

³⁵ módulo : aplicativo

³⁶ Módulo Form : aplicação on-line , contendo lógicas de atualização, telas, botões, itens, etc.

³⁷ Módulo Menu : conjunto de submenus e lógicas para acionamento de outros módulos.

³⁸ Módulo PL/SQL Library : conjunto de programas PL/SQL que podem ser compartilhado por diversas aplicações *forms*.

³⁹ Object Library : conjunto de objetos internos do *Form* que podem ser definidos uma unica vez e reutilizados em outras aplicações.

Conjunto de programas e serviços para atender a arquitetura de três camadas, fornecendo uma infraestrutura para disponibilização de aplicações voltadas para a *Web*, de tal modo que reduz a preocupação com protocolos de comunicação e serviços de rede dos desenvolvedores, liberando-os para concentração na solução dos problemas do negócio da organização. O *Forms Services* possui a capacidade de balanceamento de execução das aplicações que permite a otimização dos recursos de *hardware*. Com esta característica quando o limite da capacidade na máquina servidora de aplicação for atingido, poderão ser adicionadas mais máquinas, balanceando a execução das aplicações pelas diversas máquinas (Fernandes,2002). O *Forms Services* permite que aplicações escritas para ambiente cliente-servidor sejam visualizadas através do *browser*.

3.4.1.1 Componentes do *Forms Services*:

O *Forms* é composto pelos seguintes itens:

- . ***Forms Applet*** – Quando o usuário abre uma sessão *Forms* via *Web*, este componente é carregado dinamicamente. Ele faz a interface entre o cliente(*browser*) e o *Forms Runtime*.
- . ***Forms Listener*** – Componente responsável pela execução da conexão entre cliente e runtime no servidor de aplicações, recebem, armazenam e disponibilizam documentos HTML.
- . ***Forms Runtime Engine*** – Este componente processa a lógica das aplicações no servidor de aplicação. Em comunicação com o servidor de banco de dados age como um cliente solicitando informações.

Quando um usuário usa seu *browser*, este aciona a URL que indica uma solicitação *Forms* a ser executada, uma página inicial é enviada ao *browser*. Neste momento pode mandar ao cliente o arquivo *Java* contendo o *Forms Applet*. Neste instante o *Forms Applet* é iniciado e os parâmetros informados pelo cliente tais como, *login*⁴⁰ e outros, são capturados e indica que aplicação *Forms* deve ser iniciada. Esta envia uma requisição e os parâmetros ao *Listener* (corresponde a uma porta específica na máquina servidora de aplicação), este estabelece uma conexão entre o cliente e o *Forms Server*, após esta conexão a comunicação fica entre o *Forms Applet* e o *Forms Runtime* e entre o *Forms Runtime* e o banco de dados (Fernandes,2002) como demonstra a Figura 3.2.

⁴⁰ *Login* : usuário e senha de acesso .

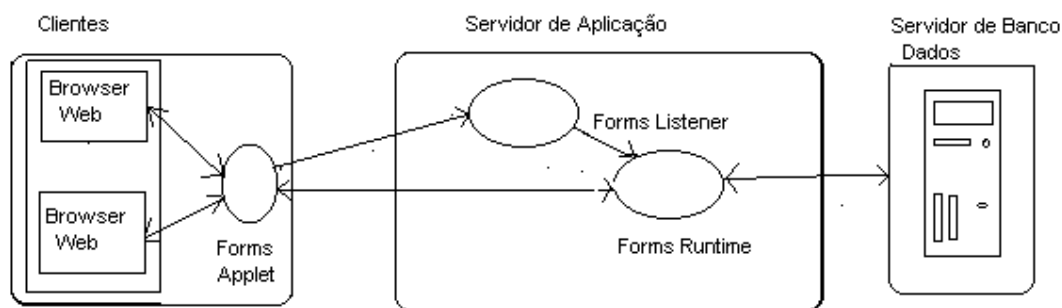


Figura 3.2 – Figura da arquitetura Web mostrando os componentes do Forms Services

3.4.2 Métodos de conexão usados pelo *Forms Services*

A comunicação entre cliente e servidor Web é feita através de uma requisição e uma resposta. O cliente envia uma mensagem pedindo abertura de conexão, e o servidor retorna uma resposta de aceite ou recusa ao pedido, caso a resposta for positiva a conexão é estabelecida. A recusa pode acontecer quando o número máximo de pedidos pendentes for atingido. Quando uma conexão for estabelecida o cliente envia uma mensagem com uma requisição do usuário, representada por uma URL. A requisição é enviada através da rede ao servidor, que após o processamento envia uma resposta ao cliente, que ao recebê-la, retorna uma requisição de fechamento de conexão (Lima,1997). Os métodos de conexão usados pelo *Forms Services* são (Fernandes,2002):

- . *Sockets* - Esta conexão utiliza uma interface padrão de comunicação TCP/IP⁴¹.
- . HTTP - Este tipo de conexão utiliza HTTP *socket* para conexão. A comunicação entre o *Form Services* e o cliente é encapsulada em pacotes HTTP.
- . HTTPS - Utiliza o mesmo mecanismo do HTTP, com a adição de regras de segurança SSL⁴², como um protocolo transparente com o objetivo de fornecer privacidade (com criptografia das mensagens), integridade (impede que as mensagens sejam modificadas) e autenticação (a máquina-cliente tem condições de verificar se o servidor é quem diz ser).

⁴¹ TCP/IP : *Transmission Control Protocol/Internet Protocol*.

⁴² SSL: secure sockets layer

3.4.3 Configuração do Browser

Os usuários poderão utilizar as seguintes configurações no browser para utilizar as aplicações *Oracle Forms* na *Web* (Fernandes,2002):

- . *Native JVM* usando *Internet Explorer 5*.
- . *Oracle Jinitiator plug-in* usando *Netscape Navigator* ou *Internet Explorer* – Define uma JVM (*Java Virtual Machine*) no cliente em vez de utilizar a JVM *default* do browser.
- . *Applet Viewer* (Componente do JDK – *Java Developer Kit*) – Nesta forma de comunicação o uso do protocolo HTTPS não é permitido.

3.4.4 Execução das aplicações *Forms*

Nas execuções de aplicativos feitos em *Forms* pode-se utilizar *Forms Servlet*⁴³ ou *Forms CGI*⁴⁴, utilizando o arquivo de configuração *formsweb.cfg* (Fernandes,2002) . O uso de *servlets* torna-se a criação dos arquivos HTML mais rápidos, que são as maiores vantagens principalmente em situações de grande tráfego na rede.

O *Forms Servlet* detecta o tipo de *browser* do cliente e gera a página HTML mais adequada, determinando as *tags*⁴⁵ corretas (Fernandes,2002).

Para que as aplicações *Forms* sejam executadas é necessário instalar um servidor *Web*, podendo utilizar qualquer servidor pessoal. Caso seja utilizado o servidor *Web Oracle9i IAS* este cria alguns caminhos virtuais para a instalação dos arquivos de software do *Forms* (Fernandes,2002).

3.4.5 Cuidados a serem tomados pelo desenvolvedor *Web*

A seguir são apresentadas algumas das características não suportadas no ambiente *Web* na programação utilizando o *Forms Developer* em aplicações desenvolvidas para cliente/servidor (Fernandes,2002).

- . Os *Triggers*⁴⁶ *When-Mouse-Enter*, *When-Mouse-Move*, *When-Mouse-Leave*⁴⁷ devem ser eliminadas, pois torna-se os aplicativos muito lentos inviabilizando a sua execução.

⁴³ *Forms Servlets* : Aplicativo que estende a funcionalidade de um servidor Web

⁴⁴ CGI : Interface gráfica do usuário

⁴⁵ *tags* : são palavras chaves delimitadas pelos símbolos “<” e “>”, que funcionam como instruções para os navegadores efetuarem uma determinada tarefa.

⁴⁶ *triggers* : é um conjunto de comandos (bloco PL/SQL) disparado por um evento específico.

. Deve-se eliminar, se possível, algumas triggers da programação do Forms, pois impacta negativamente no desempenho tais como: *When-Mouse-Click*, *When-Mouse-DoubleClick*, *When-Mouse-Down* e *When-Mouse-Up* e *When-Mouse-Move*⁴⁸.

. Características que são executadas no ambiente *server* como ActiveX⁴⁹, OCX⁵⁰, OLE⁵¹, VBX⁵² os resultados não podem ser visualizadas pelos clientes.

. Funções como: *Host*, *Ora-Ffi*, *User-Exit* são executadas no servidor de aplicação, portanto se houver a necessidade de apresentação ao cliente do resultado a aplicação deverá ser modificada.

. Botões utilizando imagens com extensão .ico deverão ser substituídos pelos .gif ou .jpg.

. Quando usar o *Run_product*⁵³ para executar um relatório, os parâmetros deverão ser passados pelo *forms*, como o relatório é executado no servidor de aplicação, a apresentação da tela de parâmetros não é visível pelo cliente.

3.4.6 Sugestões da Oracle para melhorar o desempenho do aplicativo

A seguir são apresentadas algumas sugestões para melhorar o desempenho dos aplicativos.

. O desenvolvedor deverá se preocupar com o desempenho da aplicação. Algumas características devem ser observadas tais como: a frequência de acesso ao banco de dados, com as imagens apresentadas, pois toda a vez que a imagem é apresentada é feito um *download* do servidor de aplicação ou extraída do banco de dados, deve-se possível reduzir o número de imagens, elas são baixadas no cliente a cada vez que forem apresentadas em uma tela da aplicação. A *Oracle* sugere que o logotipo da empresa seja apresentado no HTML de abertura, que é apresentado no início da execução da aplicação.

. A *Oracle* sugere a criação de uma página de HTML que funcione como um menu principal. Tendo apenas um endereço de entrada. Eliminando a necessidade de informar ou

⁴⁷ *When-Mouse-Enter*, *When-Mouse-Move*, *When-Mouse-Leave* : triggers disparadas quando da navegação pelo mouse.

⁴⁸ *When-Mouse-Click*, *When-Mouse-DoubleClick*, *When-Mouse-Down*, *When-Mouse-Up* e *When-Mouse-Move* : triggers de navegação executadas quando comando são acionados pelo mouse.

⁴⁹ ActiveX : container para controle.

⁵⁰ OCX : é uma biblioteca.

⁵¹ OLE : *Object Linking and Embedding* permite acesso a diferentes aplicações por exemplo: acesso à calculadora do *Windows* ou ao *Word*.

⁵² VBX : biblioteca desenvolvida por terceiros que podem ser anexados nas aplicações *forms*.

⁵³ *Run_product*: comando de execução de um relatório através de um aplicativo *forms*.

distribuir a URL de cada um dos novos sistemas a ser criados ou removidos. Bastando adicionar ou remover a URL nova ou indesejada do menu.

- . A utilização de *timers* na aplicação deve ser eliminada ou diminuída. Pode ser utilizada a implementação de *Java Beans*, que não requerem intervenção do *Forms Services* e da rede.

- . Colocar objetos similares, tais como: botões com botões, juntos no *Object Navigator*. Esta ordenação faz com que as propriedades enviadas para o cliente sejam feitas utilizando um algoritmo que diminui a quantidade de informações trafegadas, melhorando o desempenho.

- . Recomenda-se preencher a propriedade *prompt* do objeto ao invés de usar um *boilerplate*⁵⁴ de texto. Reduzir o uso de *boilerplates* dos tipos arco, círculos e polígonos que são mais demorados para serem carregados. Procurar utilizar os comuns, tipo retângulo e linhas.

- . Reduzir ao máximo possível as navegações entre os campos no *forms*, procurando preencher os campos com valores *default*⁵⁵. Com isto obtêm-se um ganho de performance.

- . Para os objetos que não forem visíveis, utilizar as propriedades ‘RAISE ON ENTRY=YES’ para a *Canvas*⁵⁶ e ‘VISIBLE=NO’ para que não sejam carregados. Na *Canvas* do tipo *TAB*⁵⁷ todos os elementos de todas as pastas são carregadas quando esta se torna visível. A Alternativa é dividir em várias *Canvas* do tipo *Stacked*⁵⁸.

- . Reduzir a utilização de *triggers* de validações, do tipo *When-Validate-Item*⁵⁹, pois é processada pelo *Forms Services*. Pode-se substituir com componentes *Java*, desta forma a validação do item seria enviada para o cliente, pois estaria contida no item.

- . Recomenda-se dividir grandes aplicações em várias para que o tempo de apresentação de cada tela seja reduzido.

Para coletar informações sobre o desempenho pode-se utilizar o *Forms Runtime Diagnostics*. Esta coleta as informações relativas a performance durante a execução de uma aplicação. Ajuda na depuração de aplicações, pois gera um arquivo contendo ações internas e ações externas em ordem cronológica.

⁵⁴ *Boilerplate*: gráficos que decoram a tela.

⁵⁵ *Default* : valor prédefinido ou padrão; valor assumido a não ser que seja alterado.

⁵⁶ *Canvas* : São as telas de apresentação, é onde estão os itens, textos e gráficos.

⁵⁷ *TAB* : tipo de *canvas* que permite agrupar e exibir um grande número de informações em um único *canvas*.

⁵⁸ *Stacked* : tipo de *canvas* em que os campos são acessados via *scroll*.

⁵⁹ *When-Validate-Item* : *trigger* de validação de um item, é disparado quando o valor no campo é inserido ou alterado.

3.5 Oracle Reports Developer

Oracle Reports Developer é um conjunto de ferramentas para definição, extração e geração de relatórios em arquitetura Cliente/Servidor e *Web*. Possibilitando a disseminação de informações dinâmicas nas organizações através do uso da *Intranet* e *Internet*.

O *Report Builder 6i*, é o componente principal do ambiente de construção de relatórios *Oracle Reports Developer*. Na Figura A3 do apêndice A é apresentado a tela principal do *Reports Developer*.

Fazem parte deste pacote as mesmas ferramentas que auxiliam o *Forms Builder 6i*, descritas no início da Seção 3.4 *Forms Developer* (Fernandes,2002):

. *Procedure Builder*

. *Graphics Builder*⁶⁰

. *Project Builder*

. *Query Builder*

. *Schema Builder*

. *Translation Builder*.

Com diversos auxílios do tipo de *Wizard* e integração nativa com banco de dados *Oracle*, é ainda, possível o acesso a bases que suportem *drivers* ODBC e bases ligadas em RDB, ADABAS, ou DB2 através *gateways* específicos (Fernandes,2002).

Usando uma URL é possível acionar um relatório diretamente através de um *browser*. O relatório pode ser executado através de uma página de HTML. Os relatórios gerados podem usar formato HTMLCSS⁶¹ ou PDF⁶², dispensando a instalação de diversos produtos no ambiente cliente (Fernandes,2002).

3.5.1 Reports Services

Na montagem do ambiente *Web* a arquitetura é composta de 4 componentes: um ambiente cliente, o ambiente servidor *Web*, o ambiente do *Oracle Reports Services* e o ambiente de banco de dados. Este ambiente pode tanto estar instalado em um equipamento

⁶⁰ *Graphics Builder* : Os gráficos gerados podem ser utilizados em aplicações *Reports*.

⁶¹ HTMLCSS : HTML com *cascading Style Sheets*.

⁶² PDF : *Adobe Portable Document Format*.

como em várias máquinas. Em uma arquitetura não Web não teríamos a participação do servidor Web. A diferença básica entre elas é que não há a participação do browser no ambiente cliente (Fernandes,2002).

Podemos configurar o *Reports Services* de acordo com a arquitetura utilizada, arquitetura Web ou não Web. A arquitetura Web é mais eficiente porque reduz custos de manutenção no ambiente cliente. O *Oracle Reports Services* suporta ambas as arquiteturas. Para estabelecer comunicação entre o *browser* do cliente e o *Oracle Reports Services* há necessidade de instalar e configurar o *Oracle Reports Server CGI* ou *Servlet*. O Servidor Web que determina o formato do *Oracle Reports* deverá ser CGI ou Servlet. No caso do *Oracle9i Application Server* a escolha poderá ser o formato CGI, já se o servidor Web for baseado em Java o formato deverá ser *Servlet* (Fernandes,2002). Na arquitetura não Web é preciso instalar nas máquinas cliente um software para comunicação com o ambiente servidor, isto é, o Net8 e o *Oracle Reports Thin Client*, que é composto do *Oracle Reports Launcher*, do *Oracle Reports Queue Manager* e do RWCLI60 (Fernandes,2002).

O processamento do relatório começa quando o usuário utiliza um endereço de URL, diretamente no *browser* ou através de um click em um *hiperlink*⁶³. Para processar a requisição do *browser* o servidor Web aciona o *Oracle Reports Services CGI* ou *Servlet* (dependendo da configuração), transformando em uma linha de comando que possa ser executada e inclui a solicitação em uma fila de execução. Quando o relatório for executado o *Oracle Reports Services* envia a linha de comando para execução pela *Runtime Engine*. O resultado é devolvido ao usuário através do servidor Web (Fernandes,2002).

3.6 *Oracle9i Development Suite*

O *Oracle9i Development Suite* oferece uma solução completa para criação e disponibilização de componentes Java para aplicações comerciais do mundo real. É um ambiente integrado que combina ferramentas de desenvolvimento de aplicativos e de *business intelligence*⁶⁴.

O *suite* combina além do *JDeveloper*, o *Oracle 9i* e o *Oracle Application Server*, viabilizando um ambiente de programação 100% Java. O *JDeveloper* utiliza a linguagem

⁶³ *Hiperlink*: ligação.

⁶⁴ *business intelligence* : sistemas inteligentes de estratégia de negócios para a empresa.

Java e protocolos padrão de mercado para disponibilizar a confiabilidade e escalabilidade da Plataforma *Oracle* para *Internet* (Bertini,2003).

3.6.1 Componentes do *Oracle9i Development Suite* (Oracle,2003).

. **Forms Developer**

. **Oracle9i Reports Developer**

. **Oracle9i Designer**

. **Jdeveloper**

. **Oracle9i Software Configuration Manager (Oracle9i SCM)** – Ferramenta integrada de gerenciamento de configuração de software que administra projetos.

. **Oracle9i Warehouse Builder** – Ambiente de projeto de *business intelligence* e *data warehouse* que consolida metadados e dados fragmentados de pacotes de aplicativos, aplicativos personalizados e legados, e logs da Web em um banco de dados Oracle9i.

. **Oracle9i Discoverer** - Ferramenta intuitiva de consulta, geração de relatórios e análise que permite aos usuários finais simplificar o acesso às informações.

. **Oracle9i Business Intelligence Beans** - Conjunto de componentes JavaBeans reutilizáveis e baseados em padrões, voltados à rápida implementação de avançados aplicativos analíticos em Java.

3.6.2 Jdeveloper

Um componente fundamental do *Oracle9i Development Suite*. O *Jdeveloper* é um conjunto completo de produtos de desenvolvimento para a construção de aplicações *Java* baseadas em componentes para a computação em rede. Possuem recursos para criar, depurar remotamente componentes *Java* executados em qualquer servidor com *Java Virtual Machine* padrão e implementar aplicativos baseados em componentes.

O *JDeveloper* é uma ferramenta produtiva para o desenvolvimento de aplicações *Java*, com características únicas como o *framework Oracle Business Components for Java* (Simplex,2003) . O Oracle *JDeveloper* simplifica o desenvolvimento de aplicações J2EE. É indicado para programadores *Java* e XML⁶⁵ que desejam criar aplicativos para a *Internet* e

⁶⁵ XML : *Extensible Markup Language*

para a plataforma *wireless*⁶⁶. Ambiente de desenvolvimento integrado Java, XML e SQL que possibilita o rápido fornecimento de aplicativos e Web Services (Oracle,2003).

⁶⁶ *wireless*: comunicação, transmissão de dados feita sem cabos

4. Conclusão

Em um mercado altamente competitivo as empresas estão buscando investir com planejamento e estratégia na área de informática, e com isto, consolidar o seu mercado de atuação, tornando-se mais ágeis na disseminação de informações entre seus fornecedores, filiais, distribuidores, consumidores, enfim em toda a sua cadeia de colaboradores. As empresas estão investindo pesado para integrar seus processos na *Web* buscando mais agilidade, diminuição de custos, rapidez, diferenciação dos concorrentes, enfim tornar-se mais competitivas.

Para isto, o mercado oferece uma diversidade de tecnologias disponíveis para desenvolvimento de aplicações para *Web*, fazendo com que as empresas tenham várias opções de tecnologia *Web*. Os profissionais de informática devem estar, sempre informados e capacitados para escolher a que melhor satisfaça as necessidades das empresas e usuários.

Neste trabalho o principal objetivo foi apresentar a tecnologia da *Oracle* para *Web*, o ambiente de desenvolvimento, banco de dados *Oracle* e as ferramentas que auxiliam no desenvolvimento de um sistema para *Web*. O principal produto da *Oracle* é o banco de dados *Oracle 9i*. Um banco de dados para a internet que é utilizado principalmente nas grandes empresas. As ferramentas de criação de aplicações *Forms e Reports developer*, facilitam a criação de aplicativos eficientes para *Web*, desde que os desenvolvedores sigam algumas recomendações da *Oracle*, como descrito no Item 3.4.5 do Capítulo 3 deste trabalho. Infelizmente, muitas empresas não conseguem adquirir os produtos da *Oracle* devido ao seu alto custo de aquisição, implantação e manutenção.

Através da ferramenta *Jdeveloper* a *Oracle* segue uma tendência mundial na utilização da linguagem *Java* para desenvolvimento de aplicativos para a *Web*.

Para finalizar, deve-se salientar que com a arquitetura *Oracle* para *Web* cria-se um ambiente para desenvolvimento de sistemas completo, com várias ferramentas integradas, desde o projeto do sistema com o *Oracle Designer*, criação das aplicações do sistema com o *Forms Developer*, criação de relatórios com o *Reports Developer*, desenvolvimento de aplicações *Java* com o *Jdeveloper*, o servidor Internet Application Server(IAS) e o gerenciamento do banco de dados com o *Oracle 9i*.

Referência Bibliográfica

Bertini. Disponível em : http://www.bertini.com.br/fc_Development.html . Acesso em 20 out. 2003.

Deitel,II.M. Java, como programar H. M. Deitel e P. J. Deitel; trad. Edson Furnankiewicz. 3.ed., Porto Alegre,2001.

Dorsey, Paul Dr., Hudicka Joseph R. Oracle8 Design Using UML Object Modeling. McGraw-Hill,1999.

Fernandes, Lúcia, Oracle9i Para Desenvolvedores Curso Completo. Axcel Books do Brasil Editora Ltda, 2002.

Gabriel, Glaucia, “ Estudo de Sistema de Informação para Web”, Monografia apresentada como requisito para a disciplina de Oficina de Engenharia de Software do Curso de Mestrado da UEM, Maringá, Janeiro, 2001.

Lima, Iremar, “Ambiente Web Banco de Dados: Funcionalidades e Arquiteturas e Integração”, Dissertação de Mestrado, Departamento de Informática da PUC- Rio de Janeiro, Rio de Janeiro, Agosto, 1997.

Oracle. Disponível em :<http://www.oracle.com> . Acesso em 30 out. 2003.

Simplex Solutions. Componente fundamental do Oracle9i Development Suite é o Oracle JDeveloper. Disponível em: <http://www.simplex.com.br/ids.html> . Acesso em 25 out. 2003.

Apêndice A – Telas principais das ferramentas da Oracle

A Figura A1 é tela principal do *Oracle Designer*, nela o desenvolvedor poderá construir o projeto do sistema, especificando principalmente a modelagem do sistema, definição dos dados físicos e visuais, construção dos aplicativos, geração da base de dados e armazenamento das informações importantes sobre o sistema e a base de dados.

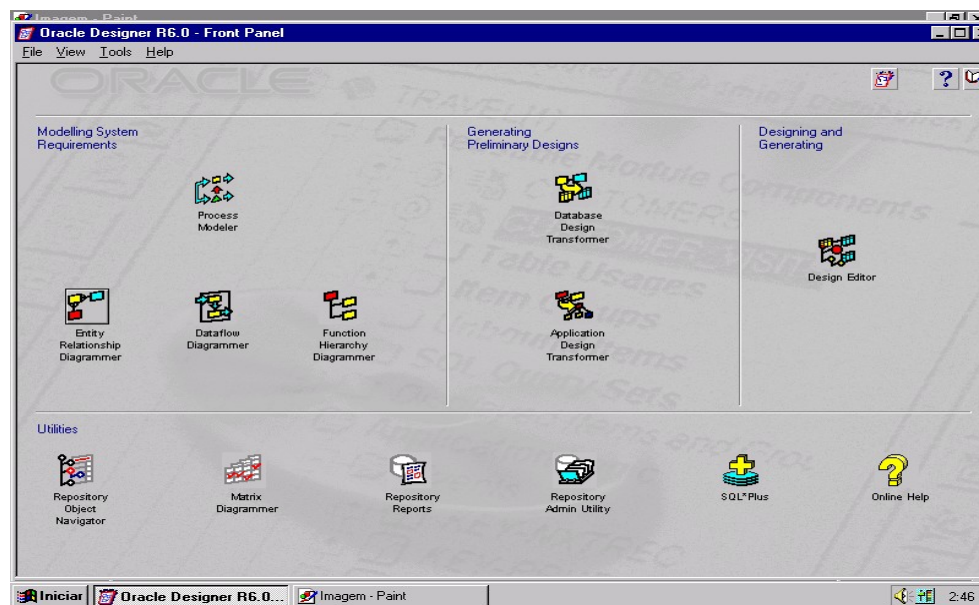


Figura A1 – Tela Principal do Oracle Designer 6.0

A Figura A2 é a tela principal da Ferramenta *Forms Developer*. Através desta tela o desenvolvedor pode criar e alterar um aplicativo que chamamos de *forms*.

A tela possui as seguintes opções:

- Menu horizontal
- Barra de Atalhos (operações do menu horizontal)
- Barra de Status
- Tópicos da árvore hierárquica
- Nós da árvore de expansão e contração.

A tela do Navegador de Objetos é uma visão hierárquica da estrutura completa que compõe o *forms builder*. Usado para criar objetos, selecionar e mover os já existentes. Quando possuir um sinalzinho de “+” a esquerda significa que possui um objeto já criado e se clicar no sinal de “+” abre-se uma estrutura mais interna e assim por diante.

Principais nós do navegador de objetos.

. *Forms* – Neste nó estará as especificações do *forms* como nome, tipo e outros.

. *Triggers* - Podemos definir uma *trigger* como sendo um conjunto de comandos (bloco *PL/SQL*) disparado por um evento específico. Uma *trigger* pode estar a nível de *forms* (será disparada em qualquer lugar do *forms*, desde que o evento seja disparado), de bloco (será disparada quando o usuário estiver posicionado somente no bloco especificado, quando este estiver em outro bloco, mesmo que ocorra o evento as *triggers* não serão disparadas) e a nível de item (será disparada quando o usuário estiver posicionado somente no item referenciado , quando este estiver em outro item, mesmo que ocorra o evento as *triggers* não serão disparadas.)

. *Alerts* - É utilizado para exibir mensagens. Aparece em uma janela e pode conter até 3 botões, dependendo do grau de gravidade da mensagem: *Stop* (parada), *Caution* (aviso) e *Note* (informativo)

. *Attached Libraries* – Propriedade do *forms* para associar uma *library* ao *forms*. Uma *library* é um conjunto de construções *PL/SQL* (*procedures, functions e packages*) compiladas. Permite compartilhar rotinas *PL/SQL*. Facilita a manutenção. Pode ser compilada independentemente dos módulos que a referênciam.

. *Data Blocks* – Estrutura básica e fundamental do *Forms Builder*, é em torno dele que todas as ações se processam, toda a funcionalidade do *forms* gira em torno do bloco. Bloco é um repositório de itens. Os itens pertencem a blocos e estão sujeitos às ações que for

especificado para o bloco que os contém. O usuário visualiza os itens , pois um bloco não tem representação visual. Um bloco pode estar ligado a uma tabela no banco de dados.

. *Canvases* – São áreas de apresentação (telas). É o lugar onde se colocam os objetos de interface, itens e os elementos de texto ou gráficos . Em uma janela podem ser apresentadas diversas *canvases*.

. *Lovs* – É uma janela contendo uma lista de valores passível de seleção, armazenados em uma estrutura chamada *record groups* (grupo de registros). Pode-se associar a *Lov* a um ou mais itens da aplicação.

. *Object Group* – Grupo de objetos tem a finalidade de facilitar o processo de cópia de diversos elementos de uma aplicação para outra.

. *Parameters* – Neste nó são especificados os parâmetros de entrada do *forms*.

. *Program Units* – Neste nó são criados os procedimentos que serão chamados dentro do *forms*.

. *Visual Attributes* – Neste nó são definidos atributos visuais que serão utilizados na definição dos itens tais como: fonte, cor, fundo e outros.

. *Windows* – É um quadro que envolve as áreas de apresentação (*canvases*). A janela faz a interface com o ambiente *Windows*.

. *PL/SQL Libraries* (Bibliotecas *PL/SQL*) – São programas *PL/SQL* armazenados na biblioteca.

. *Database Objects* (Objetos do Banco de Dados) – Neste nó serão visualizados os diversos objetos do banco de dados que o usuário terá acesso.

. *Built-in Packages* (Pacotes Embutidos) – Neste nó esta contido os diversos pacotes contendo uma variedade de rotinas que podem ser utilizadas no desenvolvimento do *forms*.

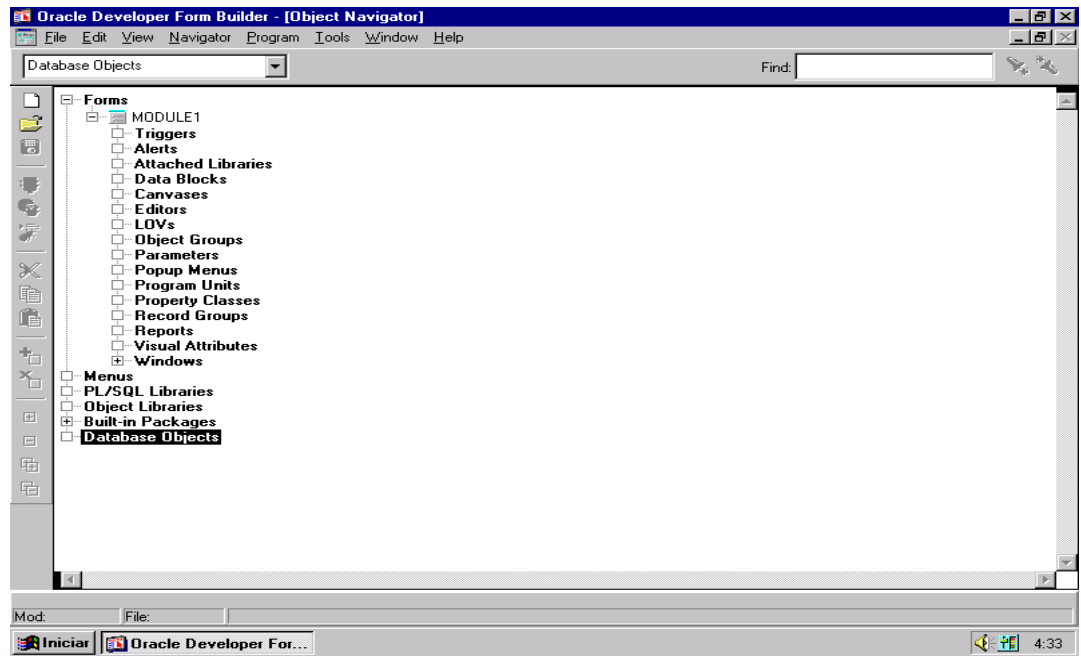


Figura A2 – Tela Principal do Forms Builder

A Figura A3 é a tela principal da Ferramenta *Report Developer*. Através desta tela o desenvolvedor pode criar e alterar um relatório.

A tela do navegador de objetos nos dá uma visão de todos os componentes da ferramenta e das aplicações criadas, e encontra-se os nós principais do navegador semelhante ao *Forms Builder* usado para criar objetos, selecionar e mover os já existentes. Quando possuir um sinalzinho de “+” a esquerda significa que possui um objeto já criado e se clicar neste sinal abre-se uma estrutura mais interna e assim por diante.

Principais nós do navegador de objetos.

. *Report* (Relatório) – Neste nó estará as especificações do relatório como nome, tipo e outros.

. *Templates* (Gabarito) – Encontra-se o nome do gabarito, isto é , algumas especificações pré-definidas em um gabarito, tais como: tamanho do campo, tipo e outras. Caso não tenha um gabarito preenchido, as especificações serão as *default* do *report*.

. *External SQL Queries* (Consultas *SQL* Externas) – Neste nó pode ser incluído comandos de *SQL Select* armazenados em arquivos, podendo ser compartilhados por outros relatórios.

. *PL/SQL Libraries* (Bibliotecas *PL/SQL*) – São programas *PL/SQL* armazenados na biblioteca.

. *Debug Actions e Stack* (Ações de depurações) – São incluídas ações temporárias, criadas para efeito de depuração.

. *Built-in Packages* (Pacotes Embutidos) – Neste nó esta contido os diversos pacotes contendo uma variedade de rotinas que podem ser utilizadas no desenvolvimento do relatório.

. *Database Objects* (Objetos do Banco de Dados) – Neste nó serão visualizados os diversos objetos do banco de dados que o usuário terá acesso.

No *Report Builder* quando um objeto puder ser criado pelo navegador, o botão criar da barra de ferramenta a esquerda ficará habilitado, caso contrário, estará desabilitado.

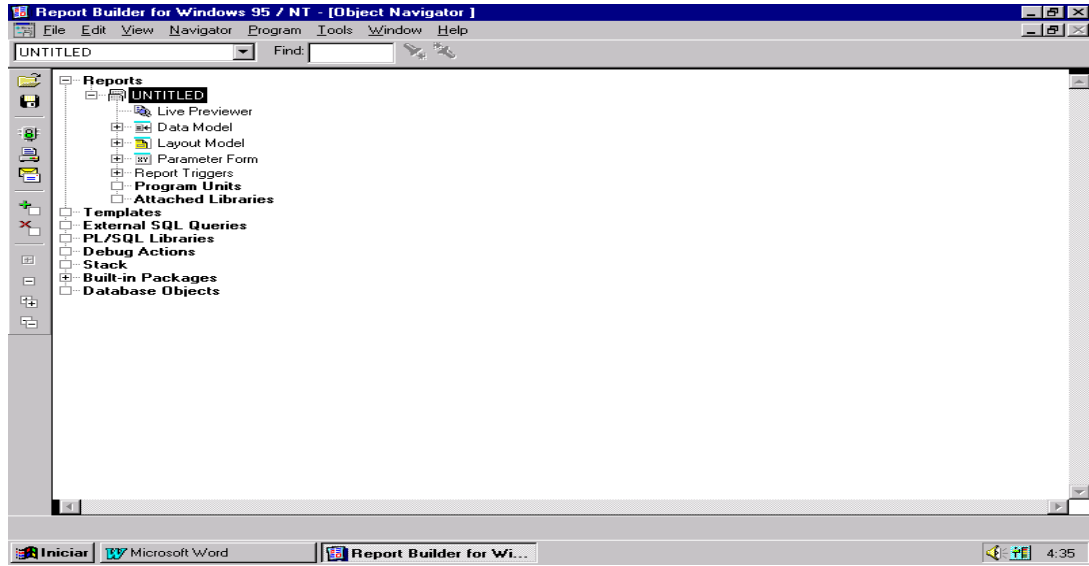


Figura A3 – Tela Principal do Report Builder

Apêndice B – Exemplos de Implementação de Recursos na Base de Dados

A Figura B1 mostra um exemplo da estrutura de um bloco *PL/SQL* que utiliza a linguagem *SQL* para selecionar registros da tabela “artigos”, cujos resultados obtidos são passados para uma variável *c1* (do tipo *Cursor*) e que utiliza comandos de controle para exibir ao usuário a partir de comandos *PL* do *Oracle*. Os cursores são como ponteiros que apontam para as colunas de uma tabela, este ponteiro deverá ser criado (*DECLARE*), aberto (*OPEN*) e executa a *query* trazendo seu resultado para uma estrutura temporária o resultado de uma pesquisa (*FETCH*) e fechado (*CLOSE*).

```

DECLARE
    Va_cod_artigo      number;
    Cursor c1 is
        Select cod_artigo
        From artigos;
BEGIN
    Open c1;
    Fetch c1 into va_cod_artigo;
    Open c1;
EXCEPTION
    WHEN CURSOR_ALREADY_OPEN
        THEN dbms_output.put_line('CURSOR ABERTO '||sqlerrm);
END;

DECLARE
    Va_cod_artigo      number;
    Cursor c1 is
        Select cod_artigo
        From artigos;
BEGIN
    Open c1;
    Fetch c1 into va_cod_artigo;
    Open c1;
EXCEPTION
    WHEN OTHERS
        THEN dbms_output.put_line('ERRO='||sqlerrm);
END;
```

Figura B1 – Exemplo de uma *PL/SQL*

No exemplo apresentado na Figura B2 é demonstrada a estrutura de um procedimento que pode ser usado em uma aplicação PL/SQL da *Oracle*.

Um Procedimento pode ser chamado a partir de outro Procedimento, Função ou de uma PL/SQL ou ainda a partir da linha do *prompt* do *SQL/PLUS* da *Oracle*.

Neste procedimento a variável *va_artigo* é definida com a mesma estrutura da coluna *des_artigo* da tabela *artigos*, isto é, com o mesmo tamanho e tipo, este tipo de definição é muito eficiente, pois caso haja alteração no tamanho ou tipo da variável na base de dados não haverá a necessidade de se alterar o procedimento. Neste procedimento é utilizado um *cursor*. Os cursores são como ponteiros que apontam para as colunas de uma tabela, este ponteiro deverá ser criado (DECLARE), aberto (OPEN) e executa a *query* trazendo seu resultado para uma estrutura temporária o resultado de uma pesquisa (FETCH) e fechado (CLOSE).

Para trabalhar cada linha de informação, caso retorne mais de uma linha na *query* deve-se utilizar os comando de *looping*. No *looping* deste exemplo é chamada uma função pré-definida do *Oracle* que mostra mensagens, neste caso os artigos e sua descrição.

```

create or replace procedure proc_artigo is
BEGIN
  DECLARE
    va_artigo artigos.des_artigo%type;
    cursor c1 is
      Select des_artigo
      From artigos;
  BEGIN
    for cont1 in c1 loop
      dbms_output.put_line('ARTIGO='||cont1.des_artigo);
    end loop;
  END;
END;
```

Figura B2 – Exemplo de uma *procedure*

A Figura B3 é um exemplo da estrutura de criação de uma Função na base de dados.

Uma Função pode receber vários parâmetros de entrada como na função abaixo e retorna apenas um.

A Função pode ser chamada a partir de outro procedimento, função ou em um comando de *PL/SQL*, pois necessita de uma variável para receber o valor de retorno, sendo assim a variável precisa ser do mesmo tipo do retorno da Função.

Na Figura B3 a função recebe o código do departamento e da seção e retorna *true* ou *false*, é feita uma pesquisa na tabela seções do registro que possua departamento e seção informados, através do atributo *sql%notfound* usado para testar se o *cursor* existe, isto é, se ele está carregado com algum valor. Retorna o valor *true* se nenhuma coluna foi selecionada, inserida, alterada ou deletada.

```
create or replace function func_secoes
    (va_cod_depto in number,
     va_cod_secao in number) return boolean is
    va_achou boolean;
    va_aux varchar2(01);
    BEGIN
        Select '1' into va_aux
        From secoes
        Where cod_depto = va_cod_depto
          And cod_secao = va_cod_secao;
        If sql%notfound
            Then va_achou := false;
            Else va_achou := true;
        End if;
        Return(va_achou);
    EXCEPTION
        when others
            Then raise_application_error(-20000,'Erro:'||sqlerrm);
    END;
```

Figura B3 – Exemplo de uma função

A Figura B4 mostra a estrutura de criação na base de dados de uma *package* agrupando uma função e uma procedure. Primeiramente é criado o cabeçalho da *package*, aonde é definido as variáveis, *procedures* e funções. A *procedure* *proc_par* recebe o valor para ser verificado se é par ou ímpar através do parâmetro de entrada (*in*) *valor*, chama a função *fun_para* que verifica se a divisão do valor por ele mesmo é igual a 0, se sim retorna verdadeiro, senão retorna falso, através deste retorno emite uma mensagem para o usuário.

```

create or replace package pac_calc is
  Function fun_par (valor in number) return boolean;
  Procedure proc_par(valor in number);
  Valor          number := 1000;
END pac_calc;

Create or replace package body pac_calc is
  Function fun_par (valor in number) return boolean is
  Aux boolean;
  BEGIN
    if mod(valor,valor) = 0
    then aux := true;
    else aux := false;
    end if;
    return(aux);
  END;
  Procedure proc_par (valor in number) is
  BEGIN
    If Fun_par(valor)
    then dbms_output.put_line('PAR');
    else dbms_output.put_line('IMPAR');
    end if;
  END;
END pac_calc;

```

Figura B4 – Exemplo de uma *package*

A Figura B5 mostra a estrutura de criação de uma *database triggers* de comando e uma de linha na base de dados. Este exemplo de triggers de comando é disparada quando um evento de deleção ocorrer na tabela de artigos. O *Before* é utilizado para que antes de ocorrer a confirmação de uma deleção na tabela artigos no banco de dados, seja enviada uma mensagem de erro e impeça a deleção por parte do usuário de qualquer registro desta tabela.

A *triggers* de linha criada na base de dados é disparada toda vez que o usuário tente deletar um registro da tabela artigos. Esta *triggers* joga para uma variável do tipo *cursor* os códigos dos artigos existentes na tabela de fornecedores que forneçam o artigo que o usuário esta tentando deletar. Caso este encontre algum registro deste relacionamento é emitido uma mensagem de erro impedindo a deleção.

Database Triggers de comando

```
create or replace trigger trig_del
Before delete on artigos
BEGIN
    raise_application_error(-20001,'Trig_del - Voce não pode deletar artigos ');
END;
```

Database Triggers de linha

```
Ex: create or replace trigger trig_delecao
Before delete on artigos
For each row
    DECLARE
        Va_aux number;
        Cursor c1 is
            Select cod_artigo
            From fornec_artigos
            Where cod_artigo = :old.cod_artigo;
    BEGIN
        Open c1;
        Fetch c1 into va_aux;
        If c1%found
            Then close c1;
                raise_application_error(-20001,'Trig_delecao - Voce não pode deletar
                    artigos . Existem dependentes');
            End if;
        Close c1;
    END;
```

Figura B5 – Exemplo de *Database Trigger*

This document was created with Win2PDF available at <http://www.daneprairie.com>.
The unregistered version of Win2PDF is for evaluation or non-commercial use only.