Universidade Federal de Pernambuco

Centro de Informática

Graduação em Ciência da Computação

XMLDB Developer – Uma ferramenta de auxílio ao aprendizado e uso do Oracle XML DB

TRABALHO DE GRADUAÇÃO

Recife, Junho de 2008

Universidade Federal de Pernambuco

Centro de Informática

Graduação em Ciência da Computação

XMLDB Developer – Uma ferramenta de auxílio ao aprendizado e uso do Oracle XML DB

TRABALHO DE GRADUAÇÃO

YURI CESAR SERAPIÃO SOARES PEREIRA

Trabalho apresentado ao Programa de Graduação em Ciência da Computação do Centro de Informática da Universidade Federal de Pernambuco como requisito parcial para a obtenção do grau de Bacharel em Ciência da Computação.

Recife, Junho de 2008

“**No fim, tudo dá certo. E, se não deu certo, é porque ainda não chegou ao fim.**”

Fernando Sabino

A todas as pessoas que, de alguma maneira, contribuíram para a formação da minha maneira de pensar, onde “maneira de pensar” pode ser lido como “caráter”, que por sua vez, pode ser entendido como “quem eu sou”.

Agradecimentos

Agradeço, sobretudo, a Deus, que, em Sua infinita misericórdia, permitiu cada passo dado, cada objetivo cumprido ao longo da minha vida. Tenho total convicção que Ele desenvolveu e usou o algoritmo mais correto para traçar o caminho ótimo no grafo de episódios durante a minha existência, tendo alguns sobressaltos mínimos locais para me livrar de outros problemas máximos globais.

Serei eternamente grato a meus pais por tudo. Jamais esquecerei todo o esforço pelo meu pai realizado para garantir a mim, e a minha irmã, alegrias momentâneas e, mais do que isso, um caráter honesto e um futuro digno. Incansável, lutou sem reclamar para nos proteger e nos criar, não obstante as várias situações difíceis às quais fomos submetidos.

Agradeço a minha primeira professora, minha mãe, que esteve sempre ao meu lado, que foi a luz que me guiou no mundo das letras, que me ensinou a ler antes mesmo que eu tivesse a oportunidade de pisar em uma escola. Sou grato ainda por suas posições firmes que modelaram meu caráter, me mostraram o caminho e por todo seu imenso carinho e amor, a mim dedicados.

Agradeço a meu avô Vicente e minha avó Olívia. Vovô, meu exemplo de homem, de grande generosidade, de caráter e honestidade irretocáveis. Ele que esteve muito presente na minha infância e assumiu um papel importante no meu crescimento. A vovó, símbolo de perseverança e dedicação ao trabalho, muito obrigado por seu amor e por suas palavras.

A minha irmã, por quem eu tenho imensa admiração, pelas palavras de carinho, de apoio e pelas brigas e discussões. A minha namorada pela paciência e pelo apoio incondicional e pelos incontáveis momentos de felicidade.

Sou muito grato aos professores do Centro de Informática da Universidade Federal de Pernambuco por terem me transmitido um pouco de seus conhecimentos, especialmente na figura do Professor Fernando Fonseca. A ele, símbolo de paixão pelo trabalho, meu muito obrigado pela atenção, pela paciência, pela tolerância, pelas oportunidades, pelas palavras amigas e pela amizade acima da relação mestre-aprendiz. Ao mestre, com carinho.

Aos meus grandes amigos da turma 2004.1 e todos os amigos do CIn que fiz de Maio de 2004 até hoje, especialmente ao grupo Mobalada. Obrigado pelos inesquecíveis momentos de alegria, descontração, conversas, futebol, bar do bigode, bar da kelly, madrugadas de projetos, e tudo mais. A saudade já é enorme e o

sofrimento pela separação, antecipado. Desejo sinceramente que esse elo inquebrável de amizade seja eterno.

A Raul Seixas que me acompanhou, dentre outros momentos, durante a realização deste. Ao Clube Náutico Capibaribe, uma paixão que supera a razão.

A todos, meu sincero e profundo **MUITO OBRIGADO.**

Resumo

XML (W3C, 1998) se tornou a tecnologia padrão para o desenvolvimento de novos formatos de documentos, atualmente (AMIANO et al., 2006). Além disso, tem sido largamente utilizada para a difusão de informações através da internet (BOURRET) e, por sua capacidade de descrever os dados que contém, tem sido usada nos mais diversos campos da Ciência (HAROLD & MEANS, 2004).

Uma ferramenta capaz de automatizar processos e auxiliar o aprendizado e uso garante maior produtividade em um ambiente de produção de software (SOMMERVILLE, 2007). O principal objetivo deste trabalho é detalhar as fases de especificação e implementação de uma ferramenta de desenvolvimento para o Oracle XMLDB, o XMLDB *Developer.*

Palavras-chave: XML, sistemas de gerenciamento de bancos de dados, ferramenta de desenvolvimento.

Sumário

Lista de figuras

1 INTRODUÇÃO

Este capítulo introduz alguns pontos significantes para a compreensão do restante do trabalho. Nele, é exposto o contexto atual no qual se insere o tema, além da motivação, do objetivo e da estrutura do documento.

1.1 CONTEXTO

A tecnologia XML (*extensible markup language*) foi desenvolvida por um grupo de trabalho dentro da W3C (*World Wide Web Consortium*) visando atingir os seguintes objetivos (W3C, 1998):

- XML deve ser diretamente usável através da Internet;
- XML deve dar suporte a uma variedade de aplicações;
- Deve ser fácil escrever programas que processem XML;
- Documentos XML devem ser legíveis por humanos e razoavelmente claros;
- O desenvolvimento em XML deve ser ágil;
- O projeto de um arquivo XML deve ser formal e conciso; e
- Documentos XML devem ser fáceis de criar.

Segundo Shanmugasundaram et al. (1999), XML está emergindo rapidamente como o padrão dominante para a representação de dados na Web. De certo modo, esta popularização se deve à capacidade de XML em descrever os dados que contém, através do uso de metadados, conhecidos no meio técnico por *tags*. Como comparação, enquanto as *tags* originais HTML (HTML40, 2008) têm como função descrever como os dados serão exibidos, as *tags* XML descrevem o significado dos dados (SHANMUGASUNDARAM et al., 2003).

A capacidade de descrever os dados contidos fez de XML um dos mais importantes elementos para a concepção da chamada Web Semântica que, de acordo

com Souza e Alvarenga (2004), não é uma web separada, mas uma extensão da atual, na qual a informação é dada com um significado bem definido.

A tecnologia XML permitiu o desenvolvimento de aplicações que exploram esse potencial de compartilhamento e exportação de informação. Entre as aplicações que têm maior destaque, estão os agregadores de conteúdo, comumente conhecidos por "leitores de *feeds*", que permitem ao usuário, centralizar e perceber as atualizações das informações que antes buscava em vários sítios diferentes.

Com a crescente popularização da tecnologia e o aumento do volume de informações descritas sob o formato XML, criou-se a demanda para engenhos capazes de armazenar e recuperar estas informações de modo eficaz. Foi com esse objetivo que a Oracle criou o 'Oracle XML DB' (ORACLE CORPORATION, 2007). O Oracle XML DB integra o armazenamento e a recuperação XML através do uso do *framework* relacional e objeto-relacional Oracle (BANERJEE et al., 2000). Ele tem a capacidade de armazenar fisicamente documentos XML, transformando-os em dados relacionais ou objeto-relacionais, e de criar documentos XML lógicos usando as funções de exportação SQL/XML (ISO/IEC 9075-14, 2003; KRISHNAPRASAD et al., 2005).

1.2 MOTIVAÇÃO

Uma grande parte dos consumidores corporativos opta pelo sistema de gerenciamento de banco de dados Oracle devido a características como robustez e desempenho. Com a inclusão do suporte à tecnologia XML no Oracle e a demanda dos consumidores, as empresas desenvolvedoras de software encaram rotineiramente o desafio de atingir um processo de desenvolvimento mais ágil, rápido, fácil e menos propenso a erros nessa tecnologia.

Por outro lado, o processo de aprendizado técnico de manipulação do Oracle XML DB apresenta algumas deficiências que ocasionam uma redução da

produtividade, dificultando o aprendizado e uso, principalmente pelos estudantes (SOMMERVILLE, 2007).

É consenso que a adoção de ferramentas capazes de automatizar etapas em um processo produtivo, contribui para o aumento de desempenho do mesmo. Tendo isso em vista, além da escassez de opções com o propósito bem definido de auxiliar profissionais e estudantes na manipulação e no aprendizado do Oracle XML DB, foi percebida a necessidade do desenvolvimento de uma ferramenta para atender estes objetivos, o XMLDB *Developer*.

1.3 OBJETIVO

Este trabalho tem como objetivo maior, explicitar todos os passos envolvidos nas especificação e no desenvolvimento de uma ferramenta capaz de auxiliar os estudantes e profissionais no aprendizado e uso do Oracle XML DB. Além deste, o trabalho objetiva apresentar o estado-da-arte em sistemas de gerenciamento de bancos de dados com suporte a XML e XML nativos, especialmente o Oracle.

1.4 ESTRUTURA

O presente trabalho estrutura-se da seguinte maneira: O capítulo 2 expõe o panorama do uso comercial do XML, SGBD e tecnologias relacionadas, além das principais técnicas de mapeamento entre os esquemas dos documentos XML e dos bancos de dados. O capítulo 2 explicita, ainda, detalhes da implementação e do uso do Oracle XMLDB. O capítulo 3, por sua vez, mostra os requisitos, casos de uso e modelagem da solução implementada, além das técnicas, processos utilizados e exemplo de uso. No capítulo 4, são apresentadas as considerações finais e sugestões de trabalhos futuros relacionados.

2 XML E SISTEMAS DE GERENCIAMENTO DE BANCOS DE DADOS

Com a necessidade de classificar o grande volume de documentos que produzia e possuía, a IBM desenvolveu, nos anos 70 (MONTEIRO, 2008), a tecnologia GML (*Generalized Markup Language*) (IBM, 2008). GML é capaz de descrever, de uma maneira simples, um documento em termos de seu formato, estrutura, conteúdo e relacionamento com outros documentos, além de outras propriedades. Os marcadores GML descrevem partes como capítulos, seções mais e menos importantes, parágrafos, listas, tabelas e assim em diante. A IBM ainda usa extensivamente a tecnologia GML para organização do enorme número de manuais que disponibiliza (IBM, 2008).

A tecnologia GML assim precedeu e foi uma das fontes usadas como base para o desenvolvimento da SGML (*Standard Generalized Markup Language*) (W3C, 2008) pela W3C (W3C, 2008). SGML é basicamente um conjunto de regras para a criação de linguagens estruturadas para a descrição de documentos. A SGML surgia então como um padrão reconhecido pela ISO para a GML, que havia atingido um alto grau de difusão já naquele momento.

A partir da SGML e com a verificação da necessidade da integração e transferência dos mais variados tipos de dados, principalmente através da *web*, esforços foram concentrados pela organização W3C para a criação de uma linguagem capaz de satisfazer os seguintes objetivos (MONTEIRO, 2008; STAKEN, 2001):

- Separação do conteúdo da formatação, ponto em que HTML é falho;
- Simplicidade e legibilidade, para humanos e computadores;
- Possibilidade ilimitada de criação de novas *tags*;
- Suporte a uma variedade de aplicações;
- Criação de arquivos de validação de estrutura;
- Integração de bancos de dados distintos; e
- Concentração na estrutura da informação, não na sua aparência.

Assim surgiu, como um subgrupo dentro da W3C, o *XML Working Group* e a tecnologia XML (W3C, 1998), que desde então tem experimentado um vertiginoso crescimento devido à sua larga adoção como tecnologia padrão pela indústria (SHANMUGASUNDARAM et al., 2003).

Nos últimos anos, XML tem sido adotado nas mais diversas áreas e com os mais diferentes propósitos. Tem sido usado como instrumento para catalogação de documentos na área jurídica e bibliotecária, como entrada de dados para simulações aeronáuticas e aplicações de análise de riscos de sistemas mecânicos, análise de crédito, análise de seguros, na robótica, multimídia, saúde, turismo, arte, construções, telecomunicações, agricultura, física e até mesmo teologia (HAROLD & MEANS, 2004). XML se tornou também a escolha mais comum no projeto de novos formatos de documentos em quase todas as aplicações de computador (AMIANO et al., 2006). *Mainframes* na *Wall Street* negociam ações através da troca de arquivos XML, jogos de computador salvam seu progresso através de XML, o envio de XML possibilitou às pessoas receberem notícias sobre os assuntos preferidos nos seus celulares (HAROLD & MEANS, 2004). XML é a sintaxe de documento mais robusta, confiável e flexível inventada até hoje (HAROLD & MEANS, 2004).

O maior volume de documentos XML, entretanto, é gerado para o transporte de informações através da *Web* (BOURRET, 2008). A maioria dos *blogs* e sítios que têm atualização constante disponibiliza o conteúdo em formato XML, permitindo deste modo que os usuários leiam, em um só lugar, seus sítios favoritos por meio dos agregadores de conteúdo. Os chamados comércio B2B (*Business to business)* e B2G (*Business to Government)*, também foram alavancados por essa revolução, uma vez que a maioria das grandes empresas e os governos de estados têm portais colaborativos para comércio que dão suporte a uma rede de fornecimento mais eficiente (LIM & WEN, 2002).

Criada a demanda, as empresas de *software* correram, naturalmente, para suprir esta carência. Tecnologias capazes de armazenar, indexar, fazer buscas de modo eficiente, processar, transformar, validar estrutura, entre outras operações, se fizeram necessárias. Ambientes de processamento como o *Saxon* (SAXONICA, 2008) e *eXist*

(EXIST, 2008), modelos de representação como o DOM (W3C, 2008), linguagens para endereçamento como o *XPath* (W3C, 2008), linguagens de consulta como o *XQuery* (W3C, 2003), engenhos de transformação com o *XSLT* (W3C, 2008), arquivos de validação de estrutura como o DTD (W3C, 2008) e o XSD (W3C, 2008) e, principalmente, a adequação dos SGBD largamente utilizados no armazenamento dos bancos de dados disponíveis através da web para dar suporte a XML.

2.1 XML BASEADOS EM DADOS

Os arquivos baseados em dados são projetados com o propósito de usar a tecnologia XML para o transporte dos dados (BOURRET, 2008; AMIANO et al., 2006; HAROLD & MEANS, 2004). Geralmente, estes arquivos são leves, até porque deverão ser enviados por uma estrutura de rede, com o objetivo final de processamento por um computador. O uso da tecnologia XML para esse fim não é usualmente um requisito primário, isto é, não é importante para a aplicação ou para o banco de dados, o fato de que os dados tenham sido por um momento armazenados em um documento XML (BOURRET, 2008). Exemplos deste tipo de XML são os arquivos que descrevem produtos em sistemas de *e-commerce*, horários de vôos nos sítios das companhias aéreas, cotação de ações e dados científicos.

Este tipo de arquivo é geralmente caracterizado por uma estrutura regular, dados finamente granularizados, isto é, unidades pequenas de dados representados, como por exemplo, um atributo e pouco conteúdo (HAROLD & MEANS, 2004). A ordem em que os elementos de um mesmo nível de profundidade aparecem não é comumente importante, exceto para o propósito de validação da estrutura (BOURRET, 2008), como mostrado no quadro 2.1.

Os arquivos XML baseados em dados têm sido usados largamente nos portais corporativos B2B e B2G, por exemplo. A maioria das grandes empresas adota um portal onde divulgam suas necessidades de produtos para estoque e fornecedores do mundo inteiro fazem ofertas para prover determinado produto. Essa comunicação é

feita através da internet, fazendo uso do XML. Essa estratégia está se tornando comum também entre órgãos dos governos de Estado, como um fator que agrega valor ao processo licitatório.

Quadro 2.1 Exemplo de um arquivo XML baseado em dados

```
<OrdemVenda OVId="12345">

      <Cliente IdCliente="543">

         <NomeCliente>Industrias ABC</NomeCliente>

         <Rua>Petrolina</Rua>

         <Cidade>Paulista</Cidade>

         <Estado>PE</Estado>

         <CodigoPostal>5000000</CodigoPostal>

      </Cliente>

      <DataPedido>981215</DataPedido>

      <Item NumeroItem="1">

         <Parte NumeroParte="123">

            <Descricao>

               <p><b>Panela Inox:</b><br />

               Aço inoxidável, base anti-aderente,

               garantia eterna.</p>

            </Descricao>

            <Preco>9.95</Preco>

         </Parte>

         <Quantidade>10</Quantidade>

      </Item>

   </OrdemVenda>
```

2.2 XML BASEADOS EM DOCUMENTOS

Ao contrário dos baseados em dados, os arquivos XML descritos nesta seção têm o propósito de representação de documentos, como livros, cartas, peças de propaganda, roteiros de filmes, por exemplo. São projetados para consumo humano. Geralmente são arquivos pesados, pelo grande volume de informação que apresentam. Estes arquivos têm uma estrutura menos regular ou até mesmo irregular, uma granularidade de dados maior, isto é, a menor unidade de dado é um conteúdo muito grande ou até mesmo o próprio documento. A ordem em que elementos do mesmo nível aparecem é sempre significante (BOURRET, 2008).

Este tipo de arquivo é geralmente usado com fins de armazenamento e posterior leitura de documentos, através de sistemas de catalogação. Não são usualmente transportados através da internet. Como exemplo, uma parte da descrição de uma peça de Shakespeare, Othello, é exibida no quadro 2.2:

Quadro 2.2 Exemplo de um arquivo XML baseado em documento

```
<PERSONAE>
<TITLE>Dramatis Personae</TITLE>

<PERSONA>DUKE OF VENICE</PERSONA>
<PERSONA>BRABANTIO, a senator.</PERSONA>
<PERSONA>Other Senators.</PERSONA>
<PERSONA>GRATIANO, brother to Brabantio.</PERSONA>
<PERSONA>LODOVICO, kinsman to Brabantio.</PERSONA>
<PERSONA>OTHELLO, a noble Moor in the service of the Venetian state.</PERSONA>
<PERSONA>CASSIO, his lieutenant.</PERSONA>
<PERSONA>IAGO, his ancient.</PERSONA>
```

```
<PERSONA>RODERIGO, a Venetian gentleman.</PERSONA>

<PERSONA>MONTANO, Othello's predecessor in the government of Cyprus.</PERSONA>

<PERSONA>Clown, servant to Othello. </PERSONA>

<PERSONA>DESDEMONA, daughter to Brabantio and wife to Othello.</PERSONA>

<PERSONA>EMILIA, wife to Iago.</PERSONA>

<PERSONA>BIANCA, mistress to Cassio.</PERSONA>

<PERSONA>Sailor, Messenger, Herald, Officers, Gentlemen, Musicians, and
Attendants.</PERSONA>

</PERSONAE>


<SCNDESCR>SCENE  Venice: a Sea-port in Cyprus.</SCNDESCR>


<PLAYSUBT>OTHELLO</PLAYSUBT>


<ACT><TITLE>ACT I</TITLE>


<SCENE><TITLE>SCENE I.  Venice. A street.</TITLE>

<STAGEDIR>Enter RODERIGO and IAGO</STAGEDIR>


<SPEECH>

<SPEAKER>RODERIGO</SPEAKER>

<LINE>Tush! never tell me; I take it much unkindly</LINE>

<LINE>That thou, Iago, who hast had my purse</LINE>

<LINE>As if the strings were thine, shouldst know of this.</LINE>

</SPEECH>
```

```
<SPEECH>

<SPEAKER>IAGO</SPEAKER>

<LINE>'Sblood, but you will not hear me:</LINE>

<LINE>If ever I did dream of such a matter, Abhor me.</LINE>

</SPEECH>



<SPEECH>

<SPEAKER>RODERIGO</SPEAKER>

<LINE>Thou told'st me thou didst hold him in thy hate.</LINE>

</SPEECH>
```

2.3 SISTEMAS DE GERENCIAMENTO DE BANCOS DE DADOS XML

Talvez a mais importante razão para o uso de XML em SGBD é a exposição de informações armazenadas em bancos de dados no formato XML (BOURRET, 2008). Por exemplo, um *web service* pode retornar em um documento XML, o preço atual de determinada ação armazenado no banco de dados da Bovespa – Bolsa de Valores de São Paulo.

Analogamente, dados podem fazer o caminho inverso e serem transferidos de uma aplicação para um banco de dados através de um documento XML (BOURRET, 2008). Por exemplo, uma instituição bancária pode submeter novas regras para empréstimo através de um documento XML. Como parte do processamento deste documento, a aplicação da instituição realiza a extração e o armazenamento destas informações num banco de dados.

Outro exemplo de uso do XML, que particularmente vem crescendo muito em popularidade, é para a modelagem de dados semi-estruturados, especialmente na área das ciências biológicas (HAROLD & MEANS, 2004). A vantagem principal de XML

neste caso é a sua extensibilidade. O campo das ciências biológicas cresce rapidamente, logo é impossível prever todos os tipos de dados disponíveis num futuro próximo. Assim sendo, um esquema relacional determinado para a área das ciências biológicas logo fica obsoleto. Ainda mais, o volume de dados nesse campo é enorme, ditando a necessidade de um banco de dados para armazenar, gerenciar e fazer consultas sobre os dados em formato XML (HAROLD & MEANS, 2004).

2.3.1 XML É, POR SI SÓ, UM BANCO DE DADOS?

É comum a confusão entre XML, por si só, e bancos de dados em ambientes profissionais de desenvolvimento na indústria de *software*. Neste sentido, um documento XML é um banco de dados no significado mais estrito da definição deste, ou seja, é uma coleção de dados inter relacionados(BOURRET, 2008). Neste caso, um documento XML não é diferente de qualquer outro documento em qualquer outro formato, até porque todos contêm dados de alguma forma (BOURRET, 2008; STEEGMANS et al., 2004).

Usado como 'banco de dados', XML tem algumas vantagens. Por exemplo, é auto-descritivo através dos seus marcadores, os quais descrevem a estrutura do documento, é portável e pode descrever os dados numa estrutura de árvore ou de grafo (HAROLD & MEANS, 2004; STEEGMANS et al., 2004). Como desvantagem principal, tem-se o longo tempo de acesso e consulta, devido ao *overhead* causado pela leitura, seleção e conversão do texto (BOURRET, 2008).

Todavia, XML e tecnologias relacionadas podem de certo modo, simular um SGBD. Alguns aspectos presentes nestes, também estão presentes em XML, como por exemplo: armazenamento, através dos arquivos, esquemas, através de DTD (W3C, 2008) e XML *Schema* (W3C, 2008), linguagens de consulta, como *XQuery* (W3C, 2003) e *XPath* (W3C, 2008), interfaces de programação, como SAX (SAX PROJECT, 2008), DOM (W3C, 2008), dentre outras. Contudo, XML não é capaz de prover funcionalidades presentes em 'verdadeiros' SGBD: armazenamento eficiente, índices,

segurança, gerenciamento de transações e integridade dos dados, acessos de múltiplos usuários, garantia da consistência do banco através de gatilhos, funções e procedimentos, além da consulta sobre vários documentos (BOURRET, 2008).

Deste modo, é possível usar XML e tecnologias relacionadas em ambientes com poucos usuários, com baixo volume de dados, e baixo desempenho. A opção pelo uso não é recomendável na maioria dos ambientes de produção, que têm muitos usuários, requisitos estritos de integridade de dados, e a necessidade de boa performance (BOURRET, 2008; BANERJEE et al., 2000). Lista de contatos, lista de sítios favoritos, seriam bons usos de XML no sentido de um banco de dados, todavia a facilidade de uso e a grande disponibilidade de SGBD gratuitos como o *MySQL* (MYSQL, 2008) e o *PostgreSQL* (POSTGRESQL, 2008), fazem da portabilidade a única vantagem em se usar XML neste sentido (BOURRET, 2008).

Os SGBD XML são geralmente classificados em duas categorias, os SGBD XML nativos e os SGBD com suporte a XML, descritos nas seções seguintes.

2.4 SISTEMAS DE GERENCIAMENTO DE BANCOS DE DADOS XML NATIVOS

O uso direto do modelo de dados XML é a característica básica de um SGBD XML nativo (STEEGMANS et al., 2004; BOURRET, 2008). Isto é, este tipo de sistema de gerenciamento de banco de dados é especialmente projetado para persistir documentos XML e usa uma série de estruturas que viabilizam tal operação (STEEGMANS et al., 2004; BOURRET, 2008).

Estas propriedades dos SGBD XML nativos são especialmente úteis em duas situações gerais (STEEGMANS et al., 2004; BOURRET, 2008; STAKEN, 2001):

- Aquelas em que o esquema dos documentos XML a serem salvos não é conhecido no tempo do projeto; e

- Aquelas em que não há um modelo no qual os dados sejam modelados de maneira ótima, como por exemplo o modelo relacional.

Todavia, a primeira situação pode ser contornada através de operações em tempo de execução em SGBD com suporte a XML. Contudo, essa estratégia é sujeita a uma alta taxa de erros e a performance é prejudicada, uma vez que os mapeamentos não são ótimos (STEEGMANS et al., 2004).

Sistemas de gerenciamento de bancos de dados XML nativos são comumente usados para gerenciar XML baseados em documentos. Isto porque permite, entre outras operações, consultas como “quais são as palavras em negrito no segundo parágrafo da terceira seção?” (BOURRET, 2008).

Os SGBD XML nativos, como os demais, também oferecem recursos como transações, segurança, acesso multiusuário, API de programação e linguagens de consulta. Alguns oferecem ainda o recurso do *rollback*, que consiste em restaurar o último estado consistente do banco de dados quando acontece algum erro em uma transação particular (BOURRET, 2008; STAKEN, 2001; STEEGMANS et al., 2004).

Embora não exista uma definição padrão para SGBD XML nativos (BOURRET, 2008; STAKEN, 2001; STEEGMANS et al., 2004), o conceito formulado por um grupo de fabricantes, chamado *XML:DB Initiative*(XMLDB INITIATIVE, 2008), é amplamente citado e aceito: Um sistema de gerenciamento de banco de dados XML nativo é tal que (BOURRET, 2008):

- Define um modelo de dados para XML. O mínimo modelo inclui elementos, atributos, texto e ordem do documento;

- Tem um documento XML como sua unidade lógica e fundamental de armazenamento – UFS; e

- Pode usar qualquer estratégia de armazenamento físico.

A estrutura que um SGBD XML nativo usa para representar internamente o documento é chamada de modelo de dados XML. Este modelo define as informações do documento que são logicamente significantes (STEEGMANS et al., 2004). Todos os modelos de dados XML devem conter, por exemplo, informações como elementos, atributos, texto, ordem no documento. Entretanto, informações como CDATA e referências a entidades, entre outras, podem ou não estar presentes em um determinado modelo, dependendo da relevância da informação à aplicação.

Os modelos de dados XML definem as diretrizes de implementação dos SGBD XML nativos. Define, ainda, a quantidade mínima de informação que deve ser armazenada e é a partir dele que as linguagens de consulta são projetadas. Cada SGBD nativo XML é livre para definir seu próprio modelo de dados porque, no momento em que os primeiros foram implementados, não havia um modelo de dados padrão (STEEGMANS et al., 2004). Assim, há SGBD baseados no *info set* (W3C, 2008), DOM, o modelo de dados *XPath* e o modelo de dados *XQuery*, entre outros.

Segundo Banerjee et al. (2000), além de Bourret (2008) e Steegmans et al. (2004), unidade fundamental de armazenamento é o menor grupo de dados que podem ser salvos de maneira lógica consistente. Por exemplo, no modelo relacional, uma linha é a unidade fundamental de armazenamento. No caso dos SGBD XML nativos, esta unidade é o documento. O tamanho deste documento é uma decisão de projeto, uma vez que o fragmento de um documento sob um elemento é potencialmente um documento XML (STEEGMANS et al., 2004), na qual deve-se levar em consideração a natureza da aplicação e do documento em si. É factível que um livro, por exemplo, possa ser dividido em vários documentos XML, porém parece absurdo que cada documento represente um parágrafo, mas parece ideal que cada um represente um capítulo, uma seção ou até uma sub-seção (BOURRET, 2008).

Isto não significa, entretanto, que os resultados de uma consulta não possam ter um tamanho menor que uma UFS. Isso seria análogo a dizer que não se pode retornar apenas o conteúdo de uma coluna em uma determinada linha de um banco de dados relacional.

Os SGBD XML nativos são livres para escolher qualquer estratégia de implementação. Estes têm sido implementados através das seguintes estratégias (STEEGMANS et al., 2004):

- Salvar os documentos XML como um todo em CLOB em um banco de dados relacional e indexar elementos individuais e valores de atributos;
- Salvar partes dos documentos XML em um conjunto fixo de tabelas (elementos, atributos, texto e assim por diante) em um banco de dados relacional;
- Processar os documentos XML e salvá-los no modelo DOM em bancos de dados orientados a objetos; e
- Processar os documentos e salvá-los em um conjunto indexado de *hash tables.*

O desempenho é um ponto que deve ser levado em consideração no momento em que se é definida a estratégia de implementação, uma vez que este é um fator de influência no desempenho. O ponto principal a ser observado neste aspecto é se os documentos são salvos por inteiro ou em partes. Os SGBD que armazenam documentos como um todo são a melhor opção em ambientes nos quais predominam as consultas que retornam documentos completos baseados em valores indexados (STEEGMANS et al., 2004), por exemplo.

2.5 SISTEMAS DE GERENCIAMENTO DE BANCOS DE DADOS COM SUPORTE A XML

Tem-se por SGBD com suporte a XML todos os SGBD que são capazes de receber documentos XML como entrada, ou fornecê-los como saída, mas que todavia seu modelo de dados interno não é o modelo de dados XML (BANERJEE et al., 2000; BOURRET, 2008; STAKEN, 2001; STEEGMANS et al., 2004). Geralmente, os SGBD

com suporte a XML usam o modelo relacional. Instâncias individuais do modelo relacional podem ser mapeadas para uma ou mais instâncias do modelo XML. A grande maioria deles inclui *software* para a transferência de dados entre os arquivos XML e eles próprios. Esses componentes de *software* podem ser internos ou externos ao SGBD (STEEGMANS et al., 2004).

É regra geral o uso dos bancos de dados com suporte a XML para documentos baseados em dados, ou seja, para fins de transporte de informação. Estes documentos possuem um ciclo de vida curto, isto quer dizer que uma vez que os dados de um documento particular são armazenados ou processados, o documento é descartado.

Os SGBD com suporte a XML retêm apenas as informações contempladas em seu modelo de dados. Então, no caso dos bancos de dados relacionais com suporte a XML, isso significa os dados e as relações hierárquicas entre os elementos do documento (STEEGMANS et al., 2004). Todas as outras informações como comentários e DTD são ignoradas (STEEGMANS et al., 2004).

2.5.1 MAPEAMENTO DO ESQUEMA DO BANCO DE DADOS PARA O ESQUEMA XML

O mapeamento entre os esquemas do banco de dados e o esquema XML é o primeiro passo para se usar um SGBD com suporte a XML. Esta é uma atividade, assim como a escrita de consultas, que geralmente é realizada em tempo de projeto do sistema (BANERJEE et al., 2000). Estes mapeamentos são da ordem de muitos-para-muitos, ou seja, é possível mapear um esquema do banco de dados para múltiplos esquemas XML. Por exemplo, um esquema de banco de dados para funcionários de uma empresa pode ser mapeado para um esquema XML de funcionários por departamento ou um esquema XML para funcionários por faixa salarial (BOURRET, 2008; STEEGMANS et al., 2004).

Uma vez que os sistemas de gerenciamento de bancos de dados com suporte a XML e as aplicações que interagem com os mesmos visam apenas processar os dados contidos nos documentos XML, é razoável que os mapeamentos entre os esquemas

XML e do banco de dados sejam baseados nos tipos de elementos, atributos e textos, omitindo assim, estruturas como referências de entidades, seções *CDATA*, padrão de codificação, comentários, instruções de processamento, entre outros (BANERJEE et al., 2000). Deste modo, uma conseqüência do uso dos SGBD com suporte a XML é a falta de garantia de que o documento resultante de uma consulta seja exatamente igual, mesmo no sentido canônico, ao documento previamente persistido no banco (BOURRET, 2008).

Os três métodos de mapeamento entre esquemas de banco de dados relacionais e XML mais populares são: mapeamento baseado em tabelas, mapeamento objeto-relacional e mapeamento através de linguagens de consulta. O mapeamento baseado em tabelas e o mapeamento objeto-relacional são particularmente importantes porque são bi-direcionais, ou seja, estes tipos de mapeamento são capazes tanto de transferir dados entre o documento XML e o banco de dados quanto no caminho contrário (STEEGMANS et al., 2004). Embora o mapeamento através de linguagens de consulta seja unidirecional, atualmente do banco para o documento XML, ele é importante porque são muito mais flexíveis do que os outros (BOURRET; STEEGMANS et al., 2004).

2.5.1.1 MAPEAMENTO BASEADO EM TABELAS[1]

O mapeamento baseado em tabelas modela o documento XML de maneira análoga ao banco de dados relacional, isto é, reproduz no documento a mesma estrutura da base de dados (BOURRET; STEEGMANS et al., 2004). Assim sendo, os dados são agrupados no documento em linhas, as linhas agrupadas em tabelas e, possivelmente, as tabelas são agrupadas em um banco de dados, como é mostrado no quadro 2.3.

Quadro 2.3 XML resultante de um mapeamento baseado em tabelas

[1] O termo "tabela" é empregado de maneira flexível. Na verdade, a tabela pode ser qualquer *result set* ou ainda, no caso da transferência de dados entre o xml e o banco, uma visão.

```
<database>
    <table>
      <row>
        <column1>...</column1>
        <column2>...</column2>
        ...
      </row>
      <row>
        ...
      </row>
      ...
    </table>
    <table>
      ...
    </table>
</database>
```

Este tipo de mapeamento é o mais comumente encontrado em *software* de transferência de dados entre o documento XML e o banco de dados relacional (STEEGMANS et al., 2004). Existe *software* que permite a completa customização do documento XML, deixando o usuário definir, entre outras coisas, os nomes dos elementos ou se o conteúdo de uma coluna será representado no XML como elemento ou atributo.

De acordo com Bourret (2008), o mapeamento baseado em tabelas é útil para a serialização de dados relacionais, tal como quando se transfere dados entre dois bancos de dados relacionais. Seu efeito colateral óbvio é que o mapeamento não pode ser usado para nenhum documento XML que não tenha o formato especificado.

2.5.1.2 MAPEAMENTO OBJETO-RELACIONAL

O mapeamento objeto relacional é usado por todos SGBD relacionais com suporte a XML e *middleware* de comunicação (BOURRET, 2008). Através deste mapeamento, um documento XML é visto como a representação de uma série de objetos serializados, onde os objetos são mapeados em tabelas, os atributos escalares são mapeados em colunas e os relacionamentos entre objetos são mapeados em relacionamentos de chave primária e chave estrangeira (BOURRET, 2008; STEEGMANS et al., 2004).

O quadro 2.4 mostra o exemplo de um documento XML contendo dados sobre um determinado departamento de uma empresa e seus funcionários. Através do mapeamento objeto-relacional, o documento XML corresponderia ao diagrama de objetos como na figura 2.1. Assim, a figura 2.2 representa o mapeamento para as tabelas do banco de dados.

2.5.1.3 MAPEAMENTO ATRAVÉS DE LINGUAGENS DE CONSULTA

Ao contrário dos métodos de mapeamento abordados nas seções anteriores, o mapeamento através de linguagens de consulta tem como vantagem, seu alto grau de flexibilidade, uma vez que não requer que o esquema XML esteja fortemente atrelado ao esquema do banco de dados (BANERJEE et al., 2000).

Atualmente, duas linguagens têm se destacado como as opções mais adotadas nos bancos de dados relacionais com suporte a XML: a linguagem *SQL/XML* e a *XQuery*.

SQL/XML é a linguagem de consulta mais importante para SGBD com suporte a XML (STEEGMANS et al., 2004). Esta linguagem amplia as funcionalidades do *SQL* adicionando funcionalidades para a criação de documentos XML a partir de dados relacionais (BANERJEE et al., 2000). As principais funcionalidades da *SQL/XML* são um tipo de dados para XML, uma série de funções escalares para a criação de *XML* (*XMLElement, XMLAttributes, XMLForest e XMLConcat*) e uma função agregada para a criação de XML (*XMLAgg*). O quadro 2.5 mostra uma consulta em *SQL/XML* cujo resultado é mostrado no quadro 2.6.

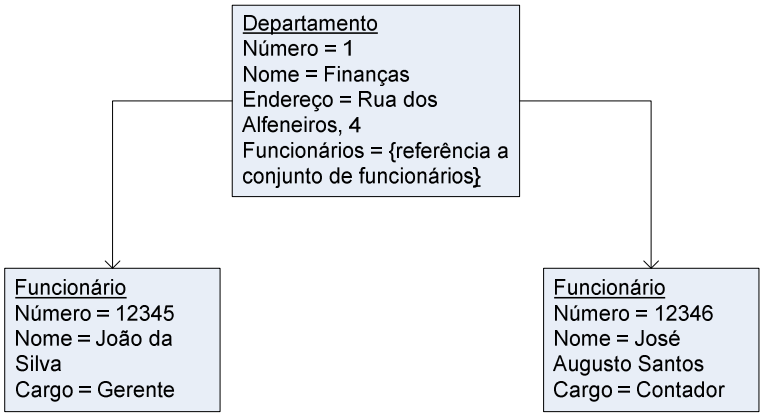

FIGURA 2.1 DIAGRAMA DE OBJETOS NO MAPEAMENTO OBJETO RELACIONAL

Número	Nome	Endereço
1	Finanças	Rua dos Alfeneiros, 4

Número	Nome	Cargo	Departamento
12345	João da Silva	Gerente	1
12346	José Augusto Santos	Contador	1

FIGURA 2.2 TABELAS DEPARTAMENTO E FUNCIONÁRIO NO MAPEAMENTO OBJETO RELACIONAL

Quadro 2.4 XML resultante do mapeamento objeto-relacional

```
<Departamento>
  <Numero> 1 </Numero>
  <Nome> Finanças </Nome>
  <Endereço> Rua dos Alfeneiros, 4 </Endereco>
  <Funcionario>
    <Numero> 12345 </Numero>
    <Nome> João da Silva </Nome>
    <Cargo> Gerente </Nome>
  </Funcionario>
  <Funcionario>
    <Numero> 12346 </Numero>
    <Nome> José Augusto Santos </Nome>
    <Cargo> Contador </Nome>
  </Funcionario>
</Departamento>
```

Quadro 2.5 Consulta na linguagem SQL/XML

```
XMLELEMENT(NAME Funcionario,
        XMLELEMENT(NAME Id, Funcionario.funcionarioId)
        XMLELEMENT(NAME Nome, Funcionario.Nome))
```

Quadro 2.6 XML resultante de consulta na linguagem SQL/XML

```
<Funcionario>
   <Id> 12345 </Id>
   <Nome> João da Silva </Nome>
</Funcionario>
```

2.6 COMPARAÇÃO ENTRE SGBD XML NATIVOS E COM SUPORTE A XML

O tipo de SGBD mais adequado a uma aplicação que tem como requisito o armazenamento e processamento de dados em documentos no formato XML é uma decisão que deve ser tomada no tempo de projeto da aplicação. Contudo, não há uma regra geral que norteie essa escolha. Alguns critérios, entretanto, devem ser levados em consideração na escolha. O desempenho esperado, a natureza da aplicação e dos documentos XML são os principais critérios que influenciam na decisão do tipo de sistema de banco de dados a ser adotado.

SGBD com suporte a XML são a melhor escolha quando os documentos XML são projetados com o propósito de transferência de informação, possivelmente fazendo uso de uma estrutura de comunicação, ou seja, documentos baseados em dados (BOURRET, 2008; STEEGMANS et al., 2004).

Desde o ano de 2003, o Banco Central do Brasil estuda a adoção do XBRL (XBRL, 2008), um padrão baseado em XML para demonstrações financeiras, como a tecnologia que daria suporte a todas as suas representações financeiras e de balanços patrimoniais, como também as que lhe são passadas pelas entidades financeiras atuantes no país (RICCIO et al., 2008).

O grupo *Twist – Treasury Group to Design Commercial Payments Standard* – cujo objetivo é desenvolver padrões para a integração de operações de comércio entre os bancos e seus clientes corporativos, atualmente trabalha no desenvolvimento de um padrão para automação do processo de pagamento comercial, baseado nas tecnologias XML e SGBD com suporte a XML (A-TEAM GROUP, 2008).

Os SGBD XML nativos são, por sua vez, recomendados em situações cujo foco é o gerenciamento de XML baseados em documentos, isto é, documentos como brochuras de propaganda, páginas da *Web*, manuais do usuário, dentre outros (STEEGMANS et al., 2004). A *Dow Jones Newswires,* por exemplo, assinou recentemente um contrato para prover um conjunto de aplicações que fornecem notícias em tempo real e informações sobre produtos para a empresa *Merrill Lynch* (A-TEAM GROUP, 2008).

Outro uso bastante comum dos SGBD XML nativos é o arquivamento de documentos. Algumas companhias, como as farmacêuticas e financeiras, por exemplo, têm que arquivar documentos por razões legais. Se esses documentos são em formato XML, então um SGBD XML nativo é uma escolha natural para tal operação (BOURRET, 2008).

2.7 ORACLE XML DB

Oracle XML DB integra o armazenamento e consulta a XML ao seu *framework* relacional e objeto-relacional (KRISHNAPRASAD et al., 2005). Assim, ele adiciona ao framework a capacidade de armazenar fisicamente os documentos XML através da

conversão dos documentos em dados relacionais e objeto-relacionais, além da criação de documentos XML lógicos através de funções de publicação SQL/XML (BANERJEE et al., 2000; KRISHNAPRASAD et al., 2005).

A Oracle desenvolveu uma técnica chamada "*XML Query Rewrite*" visando superar o desafio de processar eficientemente consultas a dados XML em um SGBD cujo armazenamento fundamental é baseado em tabelas e o engenho de consultas é orientado a tuplas (KRISHNAPRASAD et al., 2005).

Esta técnica integra as consultas a XML através de *XPath* embutido em operadores SQL e funções SQL/XML com a álgebra relacional e objeto-relacional, para isso, um conjunto de regras algébricas é usado para reduzir as consultas em equivalentes relacionais(KRISHNAPRASAD et al., 2005).

O processamento XML no Oracle XML DB é baseado no tipo de dados *XMLType*. Introduzido no Oracle 8i (BANERJEE et al., 2000), e construído sobre a estrutura objeto-relacional, o *XMLType* é similar a qualquer outro tipo de dados, tais como *Number* e *Char*, por exemplo. Operações para conversão dos dados XML em relacional e objeto-relacional, para endereçar o conteúdo do documento através de *XPath* ou para gerar documentos XML a partir de dados relacionais e objeto-relacionais são algumas das operações que o *XMLType* dá suporte.

O Oracle XML DB permite também o chamado armazenamento baseado em *XMLSchema*. Através deste, os usuários podem criar tabelas de *XMLType* na qual podem armazenar documentos XML de acordo com um *Schema* [2] previamente registrado no Oracle XML DB (KRISHNAPRASAD et al., 2005).

A funcionalidade de conversão de dados relacionais em valores XML lógicos através de *views* também é uma funcionalidade oferecida. Oracle XML DB dá suporte ao padrão SQL/XML, o qual permite que usuários usem funções de publicação como *XMLElement*, *XMLAgg*, *XMLForest* e *XMLConcat*. *XMLAgg* é uma importante função

[2] *Schema* se refere ao esquema do documento definido pela tecnologia XSD, não DTD.

de agregação para a concatenação de um conjunto de linhas de valores XML através de uma consulta relacional (KRISHNAPRASAD et al., 2005).

Oracle XML DB provê ainda uma série de funções, tais como *Extract*, *ExistsNode* e *ExtractValue* para fazer consultas a tabelas ou visões, as quais usam *XPath* para localizar e extrair dados de documentos XML. É possível ainda usar a função *XMLSequence* para converter uma lista de elementos XML em múltiplas linhas (BANERJEE et al., 2000).

3 XMLDB *DEVELOPER*

O XMLDB *Developer*, cujo desenvolvimento é a razão principal deste trabalho, objetiva se tornar a principal ferramenta de auxílio no desenvolvimento e aprendizado do Oracle XML DB.

A partir de dificuldades identificadas ao longo de meses de experiência no uso do Oracle XML DB, além de relatos de alunos [3] do Centro de Informática da Universidade Federal de Pernambuco sobre o assunto, como também através de técnicas como análise de concorrentes e prototipagem descritas a seguir, foi levantado um conjunto inicial de requisitos funcionais do XMLDB *Developer*.

Tão importante quanto, são os requisitos não funcionais. Requisitos como usabilidade e tempo de resposta são particularmente essenciais para que a aplicação atinja o grau esperado de satisfação dos usuários.

3.1 ANÁLISE DE CONCORRENTES

Como principais concorrentes do XMLDB *Developer*, foram identificadas duas ferramentas: Oracle SQL *Developer* (ORACLE, 2006) e o terminal de linha de comando do Oracle. O primeiro foi identificado devido à sua alta popularidade, além de ser um *software* que oferece um alto número de funcionalidades aos usuários. Já o segundo foi escolhido por ser a interface padrão entre os usuários e o SGBD.

[3] Os alunos foram monitorandos da disciplina Gerenciamento de Dados e Informações dos cursos de Ciência e Engenharia da Computação da UFPE, nos períodos 2007.2 e 2008.1.

3.1.1 TERMINAL DE LINHA DE COMANDO ORACLE

O terminal de linha de comando é a principal interface de interação entre os usuários e o SGBD Oracle. Embora seja uma ferramenta primitiva, ainda possui um alto grau de adoção por parte dos usuários. A partir da análise funcional do terminal, foram destacadas as seguintes características:

- É distribuído e instalado juntamente com o SGBD, o que contribui para a sua popularidade;
- Não há interface gráfica, ou seja, todas as operações, até mesmo as mais simples, como efetuar conexão (ver figura 3.1) são realizadas através da digitação de comandos, o que força o usuário a decorar os comandos ou consultar constantemente o manual;
- Cada linha digitada de um programa PL/SQL é armazenada no *buffer* até que, ao fim do programa, este seja submetido de uma vez ao SGBD. Assim sendo, a usabilidade da edição fica comprometida uma vez que uma linha digitada com erro não pode ser editada antes que o usuário submeta o programa;
- Um *script* previamente submetido não pode ser recuperado por inteiro, apenas linha por linha, através da tecla “seta para cima”. Desta maneira, o usuário é forçado a um trabalho desnecessário e cansativo;
- Ainda que o usuário opte por editar seu *script* em um editor de texto comum, o terminal dificulta as operações “copiar e colar”, uma vez que requer uma seqüência de ações não intuitivas por parte do usuário; e
- Os erros de compilação do programa PL/SQL não são mostrados no momento da submissão do mesmo. Para ver os erros de compilação de um programa previamente submetido, o usuário deve realizar outro comando, o ‘*show errors’,* como é ilustrado na figura 3.2.

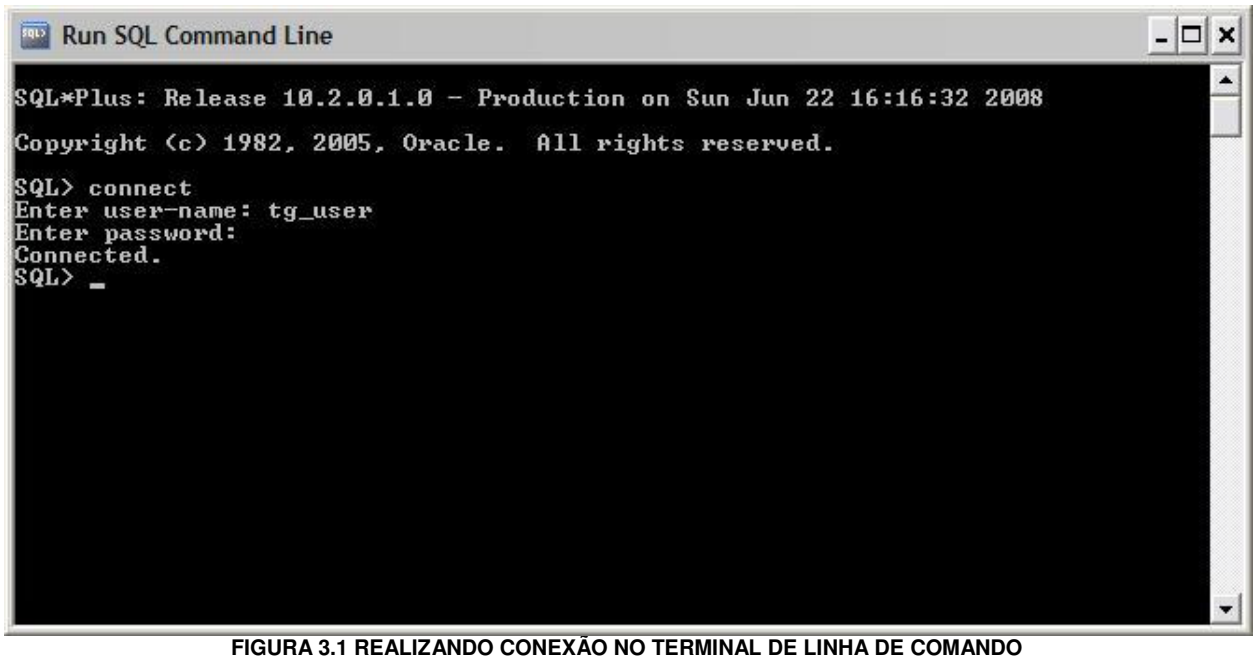

FIGURA 3.1 REALIZANDO CONEXÃO NO TERMINAL DE LINHA DE COMANDO

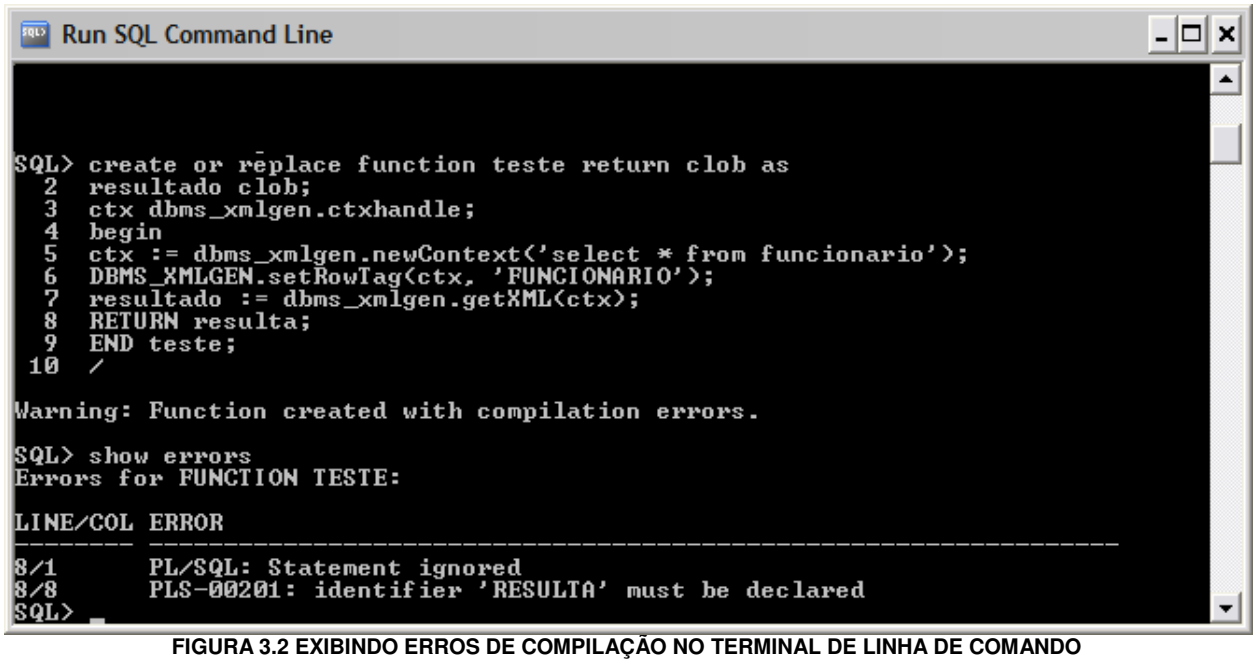

FIGURA 3.2 EXIBINDO ERROS DE COMPILAÇÃO NO TERMINAL DE LINHA DE COMANDO

3.1.2 ORACLE SQL *DEVELOPER*

O Oracle SQL *Developer* é uma ferramenta gráfica, disponibilizada livremente pela Oracle, cujo objetivo é aumentar a produtividade e simplificar as tarefas dos

desenvolvedores de projetos de bancos de dados para o SGBD Oracle (ORACLE, 2006). Desenvolvida em Java (SUN MICROSYSTEMS, 2008), o aplicativo tem a vantagem de ser independente de plataforma, podendo ser executado tanto em *Windows* (MICROSOFT, 2008) quanto em *Linux* (TLDP, 2008) e *Mac Os* (APPLE, 2008).

O Oracle SQL *Developer* é compatível com todas as versões do SGBD Oracle a partir da versão 9i, como também com as versões mais recentes dos SGBD: *Access* (MICROSOFT, 2008), *MySQL* (SUN MICROSYSTEMS, 2008) e *SQLServer* (MICROSOFT, 2008).

Os principais pontos identificados através da análise funcional do Oracle SQL *Developer* são:

- Gerenciamento de conexões – A ferramenta permite que o usuário crie, teste e salve conexões com o SGBD, como mostra o exemplo da figura 3.3. O usuário pode ainda exportar e importar essas conexões através do formato XML, funcionalidade fundamental para administradores de bancos de dados que lidam com um grande número de conexões diariamente;
- Área de execução de comandos SQL – Assim que a conexão é realizada, uma área para execução de comandos SQL é aberta automaticamente. Esta área dá suporte à criação de qualquer tipo de comando que for possível para a conexão ativa do SGBD, como por exemplo inserções SQL e funções PL/SQL, como também comandos do terminal de linha de comando. Esta área oferece ainda ao usuário, além de funcionalidades básicas como destaque de palavras reservadas, opções avançadas como execução apenas da linha selecionada ou do programa[4] como um todo, além de operações de sincronização do estado do banco de dados como *commit* e *rollback,* histórico de *scripts* executados e abas avançadas como uma que exibe a explicação do plano de execução do comando,

[4] Neste contexto, 'programa' se refere a um programa PL/SQL ou a um *script* SQL.

como pode ser visto na figura 3.4. O usuário pode abrir várias abas dessa área, podendo assim trabalhar em mais de um programa durante um determinado espaço de tempo;

- Exploração dos objetos do banco de dados – O Oracle SQL *Developer* possui a funcionalidade de exploração básica dos objetos do banco de dados (ver figura 3.5) como também uma exploração mais detalhada de um objeto, como por exemplo uma tabela dada na figura 3.6;
- Desenvolvimento PL/SQL – O aplicativo inclui um completo editor para programas PL/SQL, como pode ser notado na figura 3.7, dispondo inclusive de funcionalidades como destaque customizável de palavras reservadas, auto-compleção de código, procura e substituição de palavras no programa, além de um depurador completo e uma janela para a execução dos programas que inclusive permite ao usuário, editar o valor dos parâmetros e variáveis;
- Construtor visual de consultas – O aplicativo permite ao usuário construir consultas através de interface gráfica, utilizando apenas o *mouse*. Para isso, o usuário deve selecionar as tabelas e as colunas das tabelas nas quais a consulta será feita (ver figura 3.8); e
- Usabilidade – O SQL *Developer* tem um uso bastante intuitivo e dispõe de atalhos para a maioria das funções.

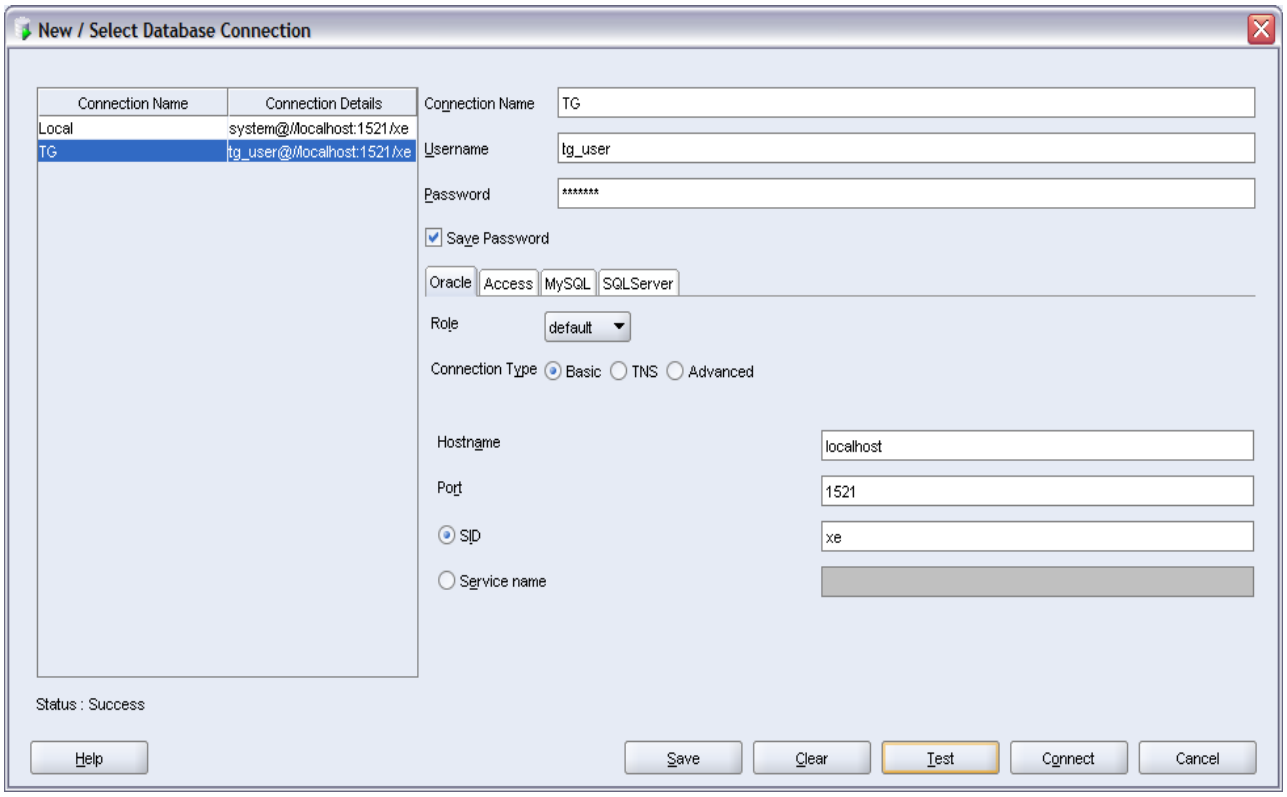

FIGURA 3.3 JANELA DE GERENCIAMENTO DE CONEXÕES

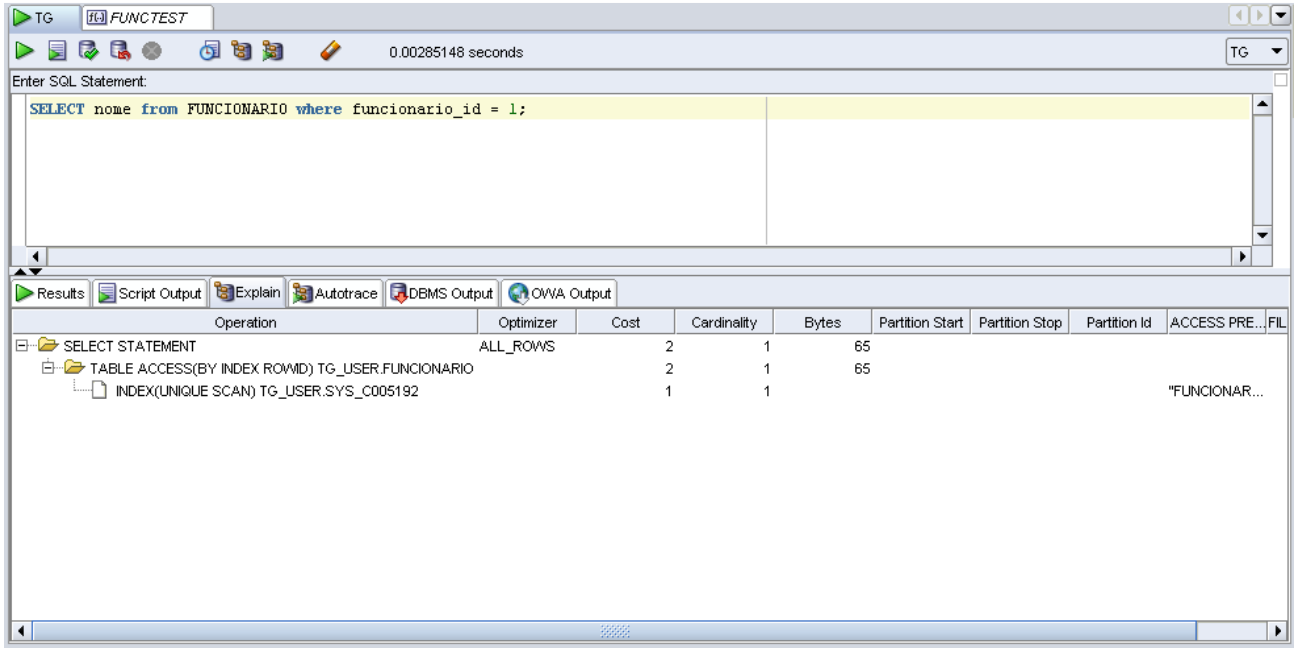

FIGURA 3.4 ÁREA PARA EXECUÇÃO DE *SCRIPTS* SQL

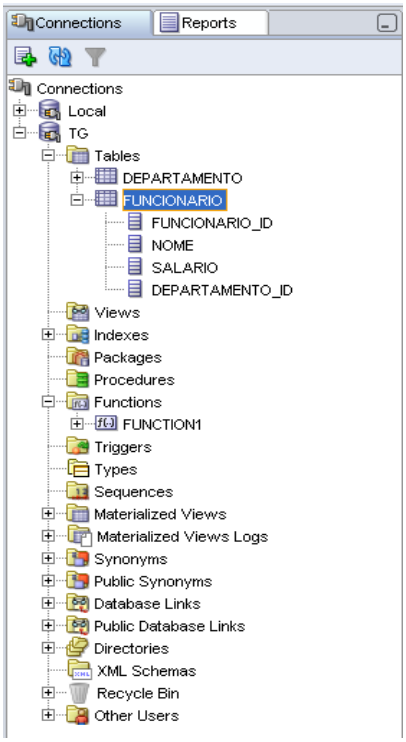

FIGURA 3.5 EXPLORADOR DE OBJETOS

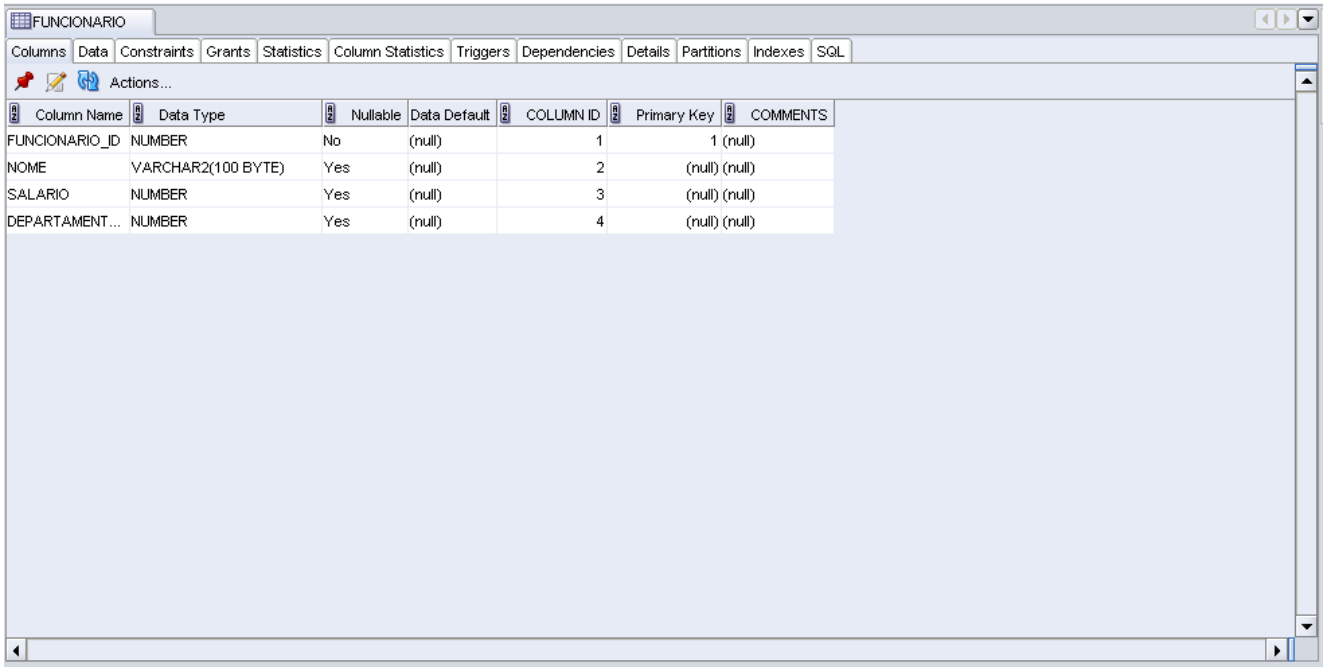

FIGURA 3.6 EXPLOREDOR DE OBJETOS DETALHADO

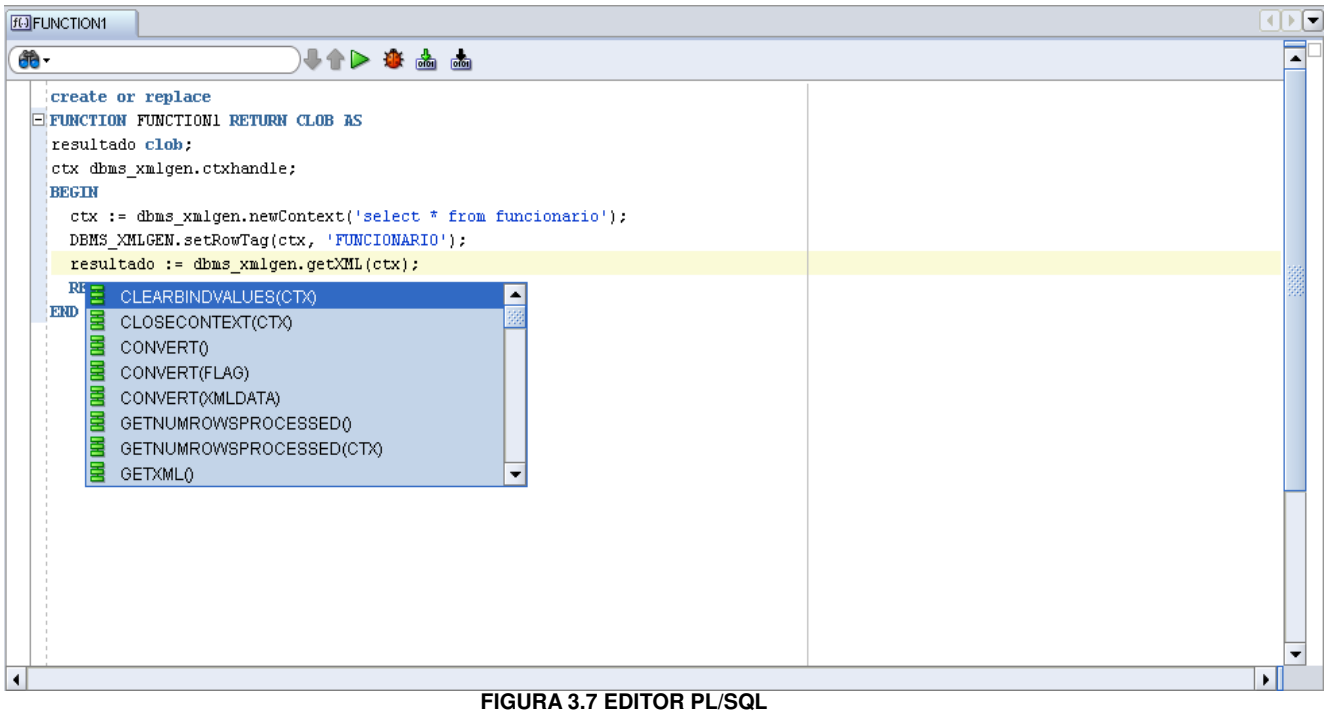

FIGURA 3.7 EDITOR PL/SQL

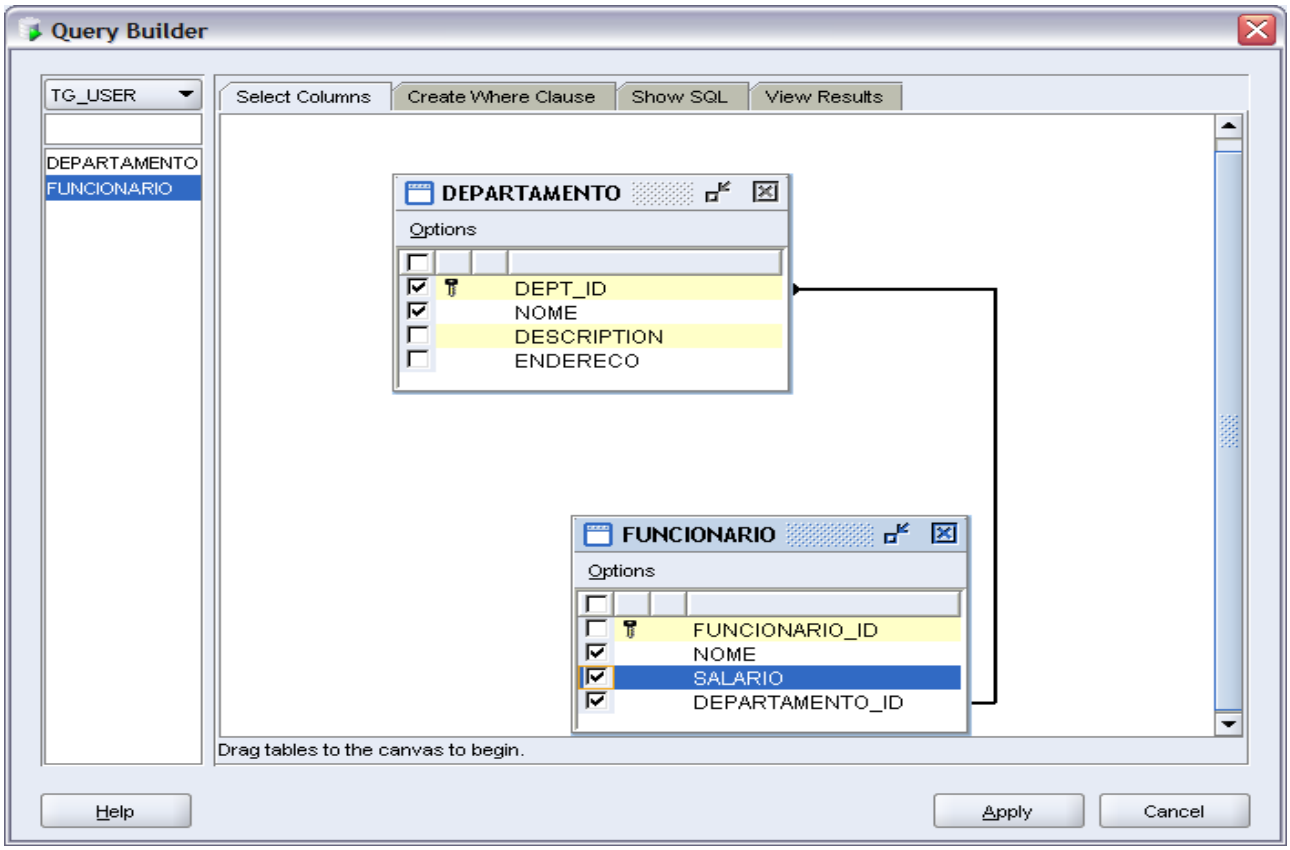

FIGURA 3.8 CONSTRUTOR VISUAL DE CONSULTAS

3.2 REQUISITOS

A partir dos pontos fortes e fracos identificados através da análise de concorrentes, foi identificado um conjunto inicial de requisitos para o XMLDB *Developer*, apresentados nas seções dadas a seguir.

3.2.1 REQUISITOS FUNCIONAIS

Foram identificados os seguintes requisitos funcionais:

- [RF-01] Gerenciador de conexões – O XMLDB *Developer* deve permitir ao usuário, a criação, o armazenamento, a deleção e o teste de conexões;
- [RF-02] Explorador de tabelas – O XMLDB *Developer* deve ser capaz de explorar as tabelas e visões de maneira simples e intuitiva, possibilitando ao usuário, visualizar as colunas e restrições (colunas *not null*, *unique*, chaves primárias e estrangeiras);
- [RF-03] Desenvolvimento SQL e PL/SQL – O aplicativo deve incluir uma área de edição de *scripts* SQL e programas PL/SQL com destaque configurável de palavras reservadas e recurso de sugestão e auto-compleção de tipos e funções do Oracle, dada a importância deste recurso que contribui para o aumento da produtividade, uma vez que livra o usuário do esforço susceptível a erros que é a digitação. O usuário deve ainda ser capaz de compilar e executar os *scripts* e programas através de um botão na interface e de atalhos do teclado. Os erros de compilação devem ser exibidos na interface gráfica, no momento em que um programa PL/SQL é compilado, sem que o usuário precise interagir para tal;

- [RF-04] Carga de arquivo XML – O XMLDB *Developer* deve permitir ao usuário, a carga do conteúdo de um arquivo XML, uma vez que essa funcionalidade deve facilitar a inserção em uma coluna *XMLType*;
- [RF-05] Armazenamento de XML – O XMLDB *Developer* deve ser capaz de salvar um arquivo XML com o conteúdo resultado de uma consulta ou execução de um programa PL/SQL;
- [RF-06] Modelo de programas PL/SQL – O aplicativo deve prover uma série de modelos de programas PL/SQL, como blocos, funções e procedimentos, para geração de arquivos XML, uma vez que estes são geralmente semelhantes em estrutura, permitindo ao usuário, especificar a consulta e o *row tag*; e
- [RF-07] Histórico de *scripts* e programas – O aplicativo deve incluir uma funcionalidade que salve automaticamente o histórico de *scripts* e programas executados pelo usuário;

3.2.2 REQUISITOS NÃO FUNCIONAIS

Além dos requisitos funcionais descritos na seção anterior, a usabilidade do XMLDB *Developer* também foi identificada como requisito não funcional primordial para o sucesso do aplicativo. As funcionalidades devem ser implementadas de modo que as ações sejam intuitivas e que diminuam, tanto quanto for possível, a necessidade de intervenção humana, além de que atalhos, no teclado, para as funções do sistema devem ser disponibilizados para os usuários.

3.3 CASOS DE USO

Os requisitos do XMLDB *Developer* foram modelados através dos casos de uso, como mostra a figura 3.9. Os casos de uso representam as funcionalidades que o

sistema incluirá em sua primeira versão. Cada caso de uso terá uma prioridade classificada em essencial, importante ou desejável. Um caso de uso essencial significa que ele é imprescindível ao sistema, ou seja, sem ele, o sistema não é capaz de realizar as tarefas a que se propõe. O sistema, embora sofra um impacto significante, é capaz de cumprir seus propósitos sem um caso de uso classificado como importante. Um caso de uso classificado como desejável pode ter sua implementação postergada para outras versões do sistema, uma vez que este não desempenha um papel fundamental no aplicativo.

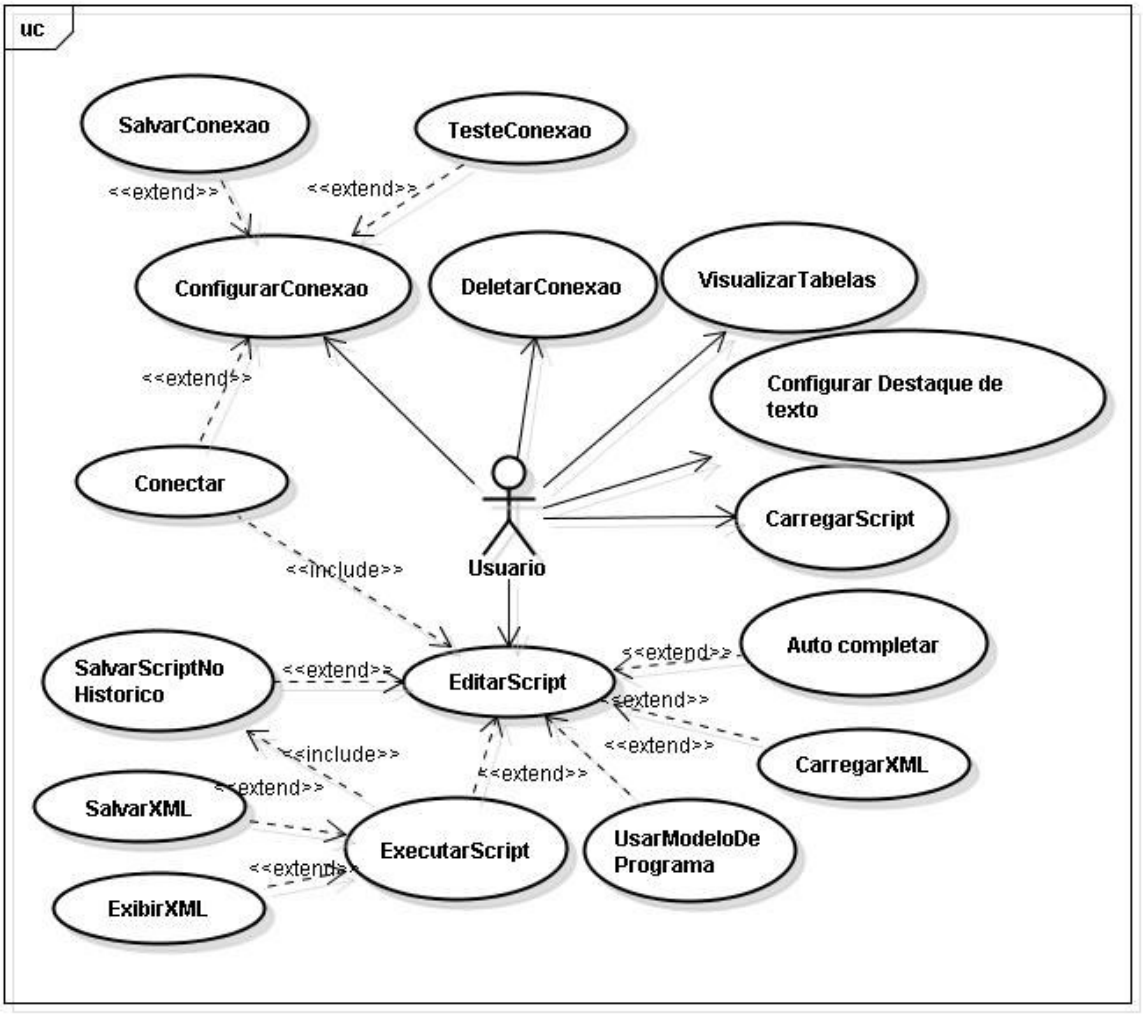

FIGURA 3.9 DIAGRAMA DE CASOS DE USO DO XMLDB *DEVELOPER*

3.3.1 [UC01] CONFIGURAR CONEXÃO

Prioridade: Essencial

Ator: Usuário

Descrição: Este caso de uso permite que o usuário configure uma nova conexão com o SGBD.

Pré-condições: Não há.

Pós-condições: É gerada uma *string* de conexão com sistema de banco de dados.

Fluxo Principal:

1. O usuário seleciona a opção 'Nova conexão';
2. O sistema exibe os campos de configuração da nova conexão;
3. O usuário insere os dados nos seguintes campos[5] relativos à nova conexão:
    - Nome da conexão;
    - Nome de usuário;
    - Senha;
    - Nome do *host;* e
    - Número da porta.

Fluxos Alternativos:

[FA01]

Caso algum dos campos não esteja preenchido, uma mensagem de erro deve ser exibida alertando ao usuário da obrigatoriedade do campo.

Pontos de extensão:

- Após o terceiro passo, o caso de uso pode ser estendido pelos seguintes casos de uso: [UC02], [UC03] e [UC04].

[5] Todos os campos são de preenchimento obrigatório

3.3.2 [UC02] CONECTAR

Prioridade: Essencial

Ator: Usuário

Descrição: Este caso de uso permite que o usuário realize uma conexão pré-configurada com o SGBD.

Pré-condições: Não há.

Pós-condições: Não há.

Fluxo Principal:

1. O usuário seleciona uma conexão do conjunto de conexões salvas pelo sistema ou a conexão atual, como extensão do caso de uso [UC01];
2. O usuário seleciona a opção 'conectar';
3. O sistema exibe o explorador de objetos com as tabelas do banco de dados;
4. **include** [UC07].

Fluxos Alternativos:

[FA01]

Se a conexão não puder ser estabelecida, uma mensagem de erro apropriada deve ser exibida para o usuário.

3.3.3 [UC03] SALVAR CONEXÃO

Prioridade: Importante

Ator: Usuário

Descrição: Este caso de uso permite que o usuário salve uma conexão pré-configurada com o SGBD.

Pré-condições: Não há.

Pós-condições: O sistema adiciona a conexão ao conjunto de conexões persistentes.

Fluxo Principal:

1. Como extensão do [UC01], o usuário seleciona a opção ‘Salvar conexão’;
2. O sistema exibe a conexão na lista de conexões salvas.

Fluxos Alternativos:

[FA01]

No passo 1, se uma conexão previamente existente tiver o mesmo nome inserido pelo usuário, o sistema exibe uma mensagem de erro adequada.

3.3.4 [UC04] TESTAR CONEXÃO

Prioridade: Importante

Ator: Usuário

Descrição: Este caso de uso permite que o usuário teste uma conexão previamente configurada com o SGBD.

Pré-condições: Não há.

Pós-condições: Não há.

Fluxo Principal:

1. Como extensão do [UC01], o usuário seleciona a opção ‘testar conexão’;
2. O sistema tenta estabelecer a conexão com o SGBD;
3. Se a conexão pôde ser estabelecida com sucesso, o sistema exibe uma mensagem de *status* de sucesso, se não, exibe uma mensagem de insucesso.

3.3.5 [UC05] REMOVER CONEXÃO

Prioridade: Importante

Ator: Usuário

Descrição: Este caso de uso permite que o usuário exclua uma conexão salva do sistema.

Pré-condições: Não há.

Pós-condições: O sistema exclui a conexão da lista de conexões persistentes.

Fluxo Principal:

1. O usuário seleciona uma conexão salva;
2. O usuário seleciona a opção 'Excluir conexão';
3. O sistema remove a conexão da lista de conexões salvas.

3.3.6 [UC06] VISUALIZAR TABELA

Prioridade: Essencial

Ator: Usuário

Descrição: Este caso de uso permite que o usuário visualize os detalhes de uma tabela em particular do banco de dados.

Pré-condições: O sistema precisa estar conectado ao SGBD e deve haver pelo menos uma tabela no esquema do banco de dados.

Pós-condições: Não há.

Fluxo Principal:

1. O usuário seleciona uma tabela exibida no explorador de objetos;
2. O sistema abre o visualizador de tabelas que deve conter as seguintes informações da tabela:
    - Colunas

- Restrições (chave primária, chave estrangeira, colunas não-nulas e colunas únicas)

3.3.7 [UC07] EDITAR *SCRIPT*

Prioridade: Essencial

Ator: Usuário

Descrição: Este caso de uso permite que o usuário insira e edite um *script* na área de edição PL/SQL.

Pré-condições: Não há.

Pós-condições: Não há.

Fluxo Principal:

1. O usuário seleciona a opção 'Abrir área de edição PL/SQL';
2. O sistema abre a área de edição PL/SQL com o foco ativo;
3. O usuário edita um *script* na área de edição;
4. Se o usuário inserir uma palavra reservada da linguagem SQL ou PL/SQL, o sistema destaca essa palavra de acordo com a configuração feita pelo usuário.

Pontos de extensão: [UC08], [UC11], [UC12], [UC13], [UC14]

3.3.8 [UC08] SALVAR *SCRIPT* NO HISTÓRICO

Prioridade: Importante

Ator: Usuário

Descrição: Este caso de uso permite que o usuário salve um *script* no histórico, para que possa ser, posteriormente, reutilizado.

Pré-condições: Deve haver um *script* digitado na área de edição PL/SQL.

Pós-condições: O *script* é adicionado ao conjunto de *scripts* persistentes do sistema.

Fluxo Principal:

1. O usuário seleciona a opção salvar *script*;
2. O usuário informa o nome do *script* a ser persistido;
3. O sistema adiciona o *script* a lista de *scripts* persistentes.

Fluxos Alternativos:

[FA01]

Se não houver *script* na área de edição PL/SQL, nenhuma ação deve ser tomada pelo sistema.

[FA02]

Se o campo do nome do *script* for deixado em branco, o sistema exibe uma mensagem de erro para o usuário.

3.3.9 [UC09] CARREGAR *SCRIPT*

Prioridade: Importante

Ator: Usuário

Descrição: Este caso de uso permite que o usuário carregue um *script* previamente persistido.

Pré-condições: Não há.

Pós-condições: O *script* é carregado em memória.

Fluxo Principal:

1. O usuário seleciona a opção 'Carregar *script*';

2. O sistema exibe a lista dos nomes dos *scripts* salvos;
3. O usuário seleciona um dos *scripts* salvos;
4. O sistema carrega o *script*;
5. O sistema exibe o *script* na área de edição PL/SQL.

3.3.10 [UC10] CONFIGURAR DESTAQUE DE TEXTO

Prioridade: Desejável

Ator: Usuário

Descrição: Este caso de uso permite que o usuário configure os padrões de destaque das palavras reservadas no texto do *script*.

Pré-condições: Não há.

Pós-condições: O padrão de destaque de texto é alterado,

Fluxo Principal:

1. O usuário seleciona a opção 'Configurar destaque de texto';
2. O sistema exibe a janela de configuração;
3. O usuário configura os seguintes aspectos:
   - A cor do texto das palavras reservadas;
   - Se o texto deve aparecer em negrito; e
   - Se o texto deve aparecer em itálico.
4. O usuário confirma a operação.

3.3.11 [UC11] AUTO COMPLETAR

Prioridade: Importante

Ator: Usuário

Descrição: Este caso permite que o usuário veja uma lista de sugestões para o texto que está digitando.

Pré-condições: Deve haver algum texto previamente digitado na posição imediatamente anterior a do cursor.

Pós-condições: Não há.

Fluxo Principal:

1. O usuário seleciona a opção ‘Auto-completar’ através das teclas ‘Ctrl + barra de espaço’;
2. O sistema seleciona o trecho de texto imediatamente anterior ao cursor até o próximo caractere de espaço, no sentido inverso ao da digitação;
3. O sistema procura palavras ou expressões que casem com a palavra parcialmente digitada pelo usuário;
4. O sistema exibe a lista de sugestões se possível, se não exibe a mensagem: “Não há sugestões”;
5. O usuário seleciona a sugestão de sua preferência;
6. O sistema completa o texto previamente digitado pelo usuário com o restante da sugestão.

Fluxos Alternativos:

[FA01]

Se não houve texto parcialmente digitado após o passo 1, o sistema não executa nenhuma ação.

3.3.12 [UC12] CARREGAR XML

Prioridade: Importante

Ator: Usuário

Descrição: Este caso de uso permite que o usuário carregue o conteúdo de um arquivo XML na posição do cursor do teclado na área de edição PL/SQL. Este caso de uso é útil para inserção de XML em colunas do tipo *XMLType.*

Pré-condições: Não há.

Pós-condições: Não há.

Fluxo Principal:

1. O usuário seleciona a opção 'Carregar XML';
2. O sistema exibe a tela de navegação no diretório;
3. O usuário seleciona o arquivo cujo conteúdo deseja carregar;
4. O sistema carrega o conteúdo do arquivo entre aspas simples na posição do cursor na área de edição PL/SQL;

Fluxos Alternativos:

[FA01]

Se o arquivo não for do formato XML, o sistema exibe uma mensagem de erro.

3.3.13 [UC13] USAR MODELO DE PROGRAMA

Prioridade: Desejável

Ator: Usuário

Descrição: Este caso de uso permite que o usuário carregue um modelo de programa[6] genérico pré definido pelo sistema. Este caso de uso é importante porque os códigos de manipulação de XML são geralmente similares.

Pré-condições: Não há.

Pós-condições: Não há.

Fluxo Principal:

1. O usuário seleciona a opção 'Usar modelo';
2. O sistema exibe uma lista de modelos de programa disponíveis;
3. O usuário seleciona um dos modelos;

[6] Neste contexto, 'programa' pode ser entendido também como *script*. O termo foi usado porque é mais adequado usar 'programa' no caso da linguagem PL/SQL, situação mais comum no caso de uso, e *script* no caso da linguagem SQL.

4. O sistema exibe o modelo na área de edição PL/SQL.

3.3.14[UC14] EXECUTAR *SCRIPT*

Prioridade: Essencial

Ator: Usuário

Descrição: Este caso de uso permite que o usuário execute um *script*.

Pré-condições: A área de edição PL/SQL não pode estar em branco, isto é, deve haver algum *script* carregado.

Pós-condições: Não há.

Fluxo Principal:

1. O usuário seleciona a opção 'Executar *Script*';
2. O sistema executa o *script;*
3. **include [UC08]**
4. O sistema exibe os resultados da execução.

Fluxos Alternativos:

[FA01]

Se houver algum erro na execução do *script*, este deve ser reportado ao usuário e a execução do restante do caso de uso interrompida.

Pontos de extensão: [UC15] e [UC16]

3.3.15[UC15] SALVAR XML

Prioridade: Importante

Ator: Usuário

Descrição: Este caso de uso permite que o usuário salve o resultado XML de uma consulta em um arquivo.

Pré-condições: O resultado da consulta deve ser do tipo *XMLType,* CLOB ou *Varchar*.

Pós-condições: Não há.

Fluxo Principal:

1. O usuário seleciona a opção 'Salvar XML' referente a uma coluna de uma linha resultante de uma consulta;
2. O sistema exibe a tela de navegação no diretório;
3. O usuário seleciona a pasta e insere o nome do arquivo a ser salvo;
4. O sistema salva o conteúdo do resultado da consulta no arquivo escolhido pelo usuário.

3.3.16 [UC16] EXIBIR XML

Prioridade: Importante

Ator: Usuário

Descrição: Este caso de uso permite que o usuário visualize o conteúdo XML resultado de uma consulta previamente realizada.

Pré-condições: O resultado da consulta deve ser do tipo *XMLType,* CLOB ou *Varchar*.

Pós-condições: Não há.

Fluxo Principal:

1. O usuário seleciona a opção 'Visualizar XML' referente a uma coluna de uma linha resultante de uma consulta;
2. O sistema exibe em uma janela o conteúdo da coluna da linha.

3.4 MODELAGEM

Após a análise dos casos de uso do XMLDB *Developer,* segundo o processo de análise e projeto de sistemas da Qualiti (QUALITI, 2008), o aplicativo foi projetado de acordo com o diagrama de classes da figura 3.10.

O sistema foi modelado de acordo com a arquitetura em camadas, buscando assim separar o código de interface, a lógica de negócios e a camada de persistência. O padrão DAO (*Data Access Objects)* (GAMMA et al., 2000) foi empregado na camada dos repositórios de dados. O padrão Fachada (GAMMA et al., 2000) também foi utilizado para centralizar os pontos de acesso ao sistema.

As classes responsáveis pela interface gráfica com o usuário têm os seus nomes começando com a palavra “Tela”. Os controladores encapsulam a lógica de negócios do sistema e em cada um deles são agrupadas as operações semelhantes, em relação ao objeto do sistema que está sofrendo uma ação, como por exemplo, o controlador “ControladorConexao”, que encapsula as operações relacionadas ao gerenciamento de conexões com o SGBD.

3.5 FUNCIONAMENTO

Para ilustrar um cenário de funcionamento do XMLDB *Developer*, foi escolhido o caso de uso “Carregar XML”. No exemplo abordado, o usuário atualiza o endereço do departamento cuja chave primária tem o valor inteiro ‘1’, de acordo com os seguintes passos:

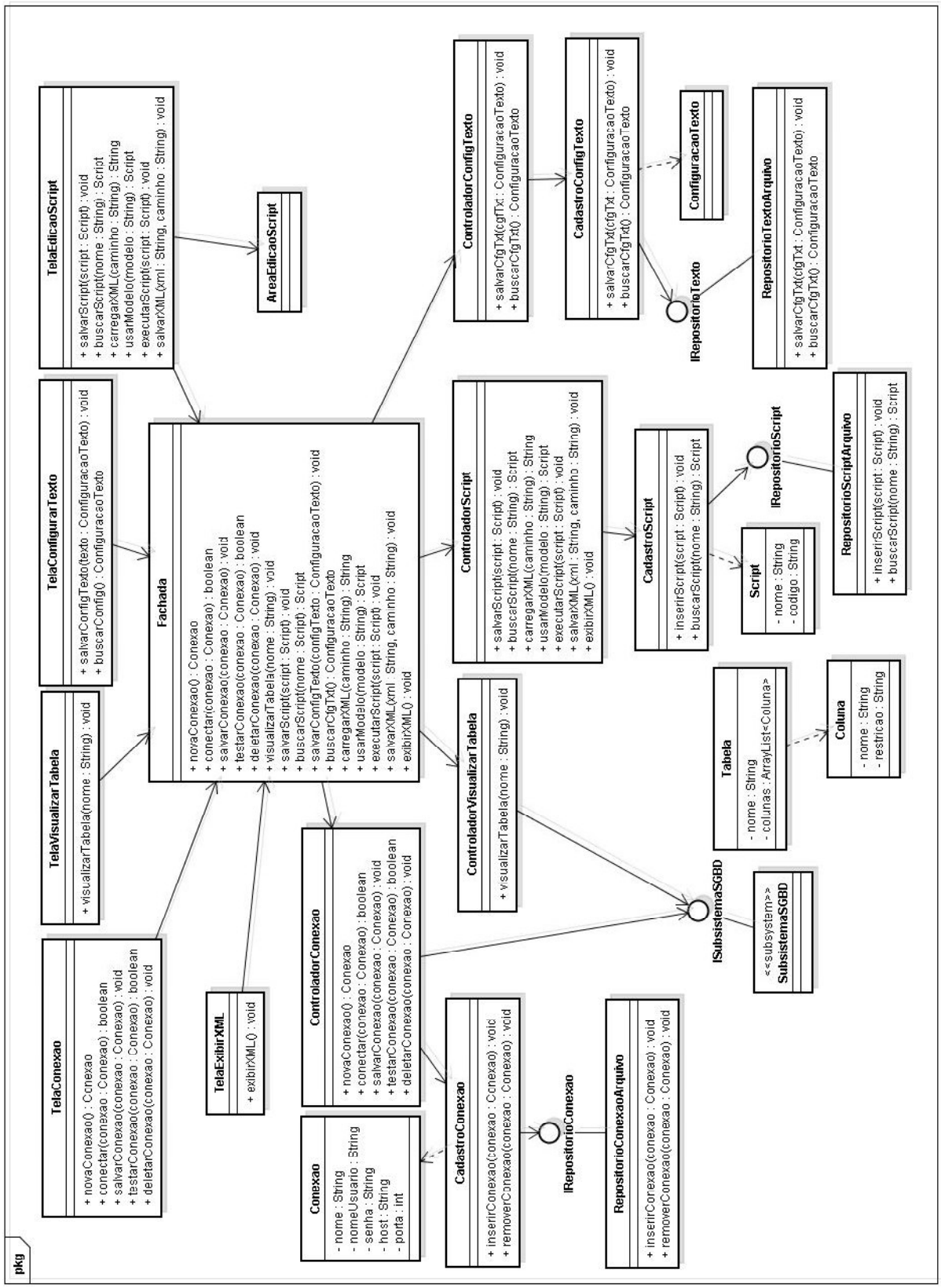

FIGURA 3.10 DIAGRAMA DE CLASSES DO XMLDB DEVELOPER

1. O usuário insere o comando a ser executado, neste caso um *Update*;
2. Quando o cursor atinge a posição do conteúdo XML, o usuário clica em "Carregar XML", como mostra a figura 3.11;
3. O sistema abre a janela de exploração de diretório, o usuário escolhe o arquivo e clica em open, como é exibido na figura 3.12;
4. O sistema carrega o conteúdo do arquivo, entre aspas simples, na posição que o cursor estava no passo 2, como ilustra a figura 3.13; e
5. o usuário completa o comando, como é mostrado na figura 3.14.

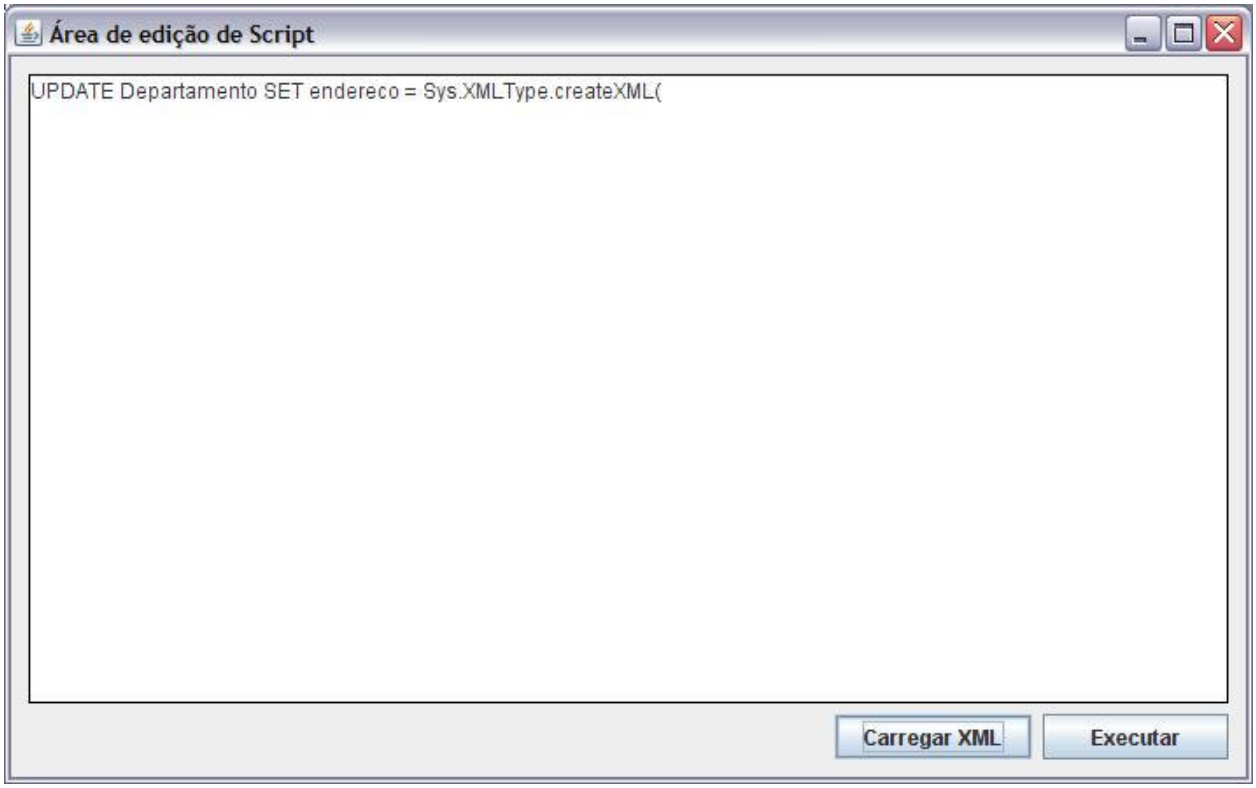

FIGURA 3.11 O USUÁRIO INSERE O COMANDO E CLICA EM CARREGAR

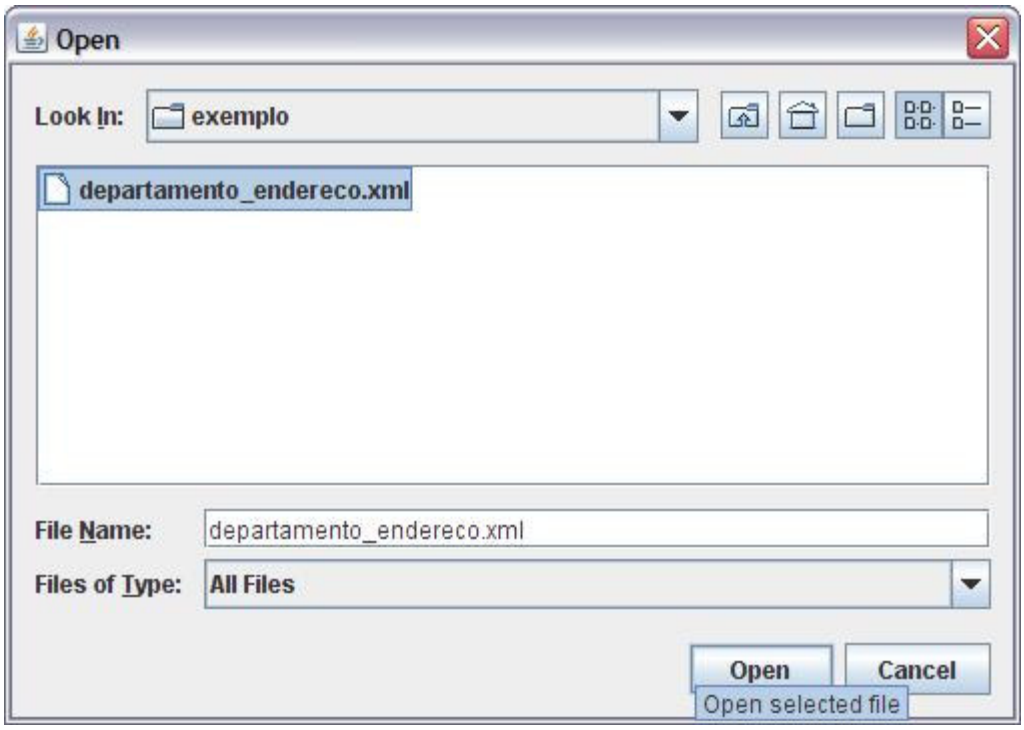

FIGURA 3.12 O USUÁRIO SELECIONA O ARQUIVO

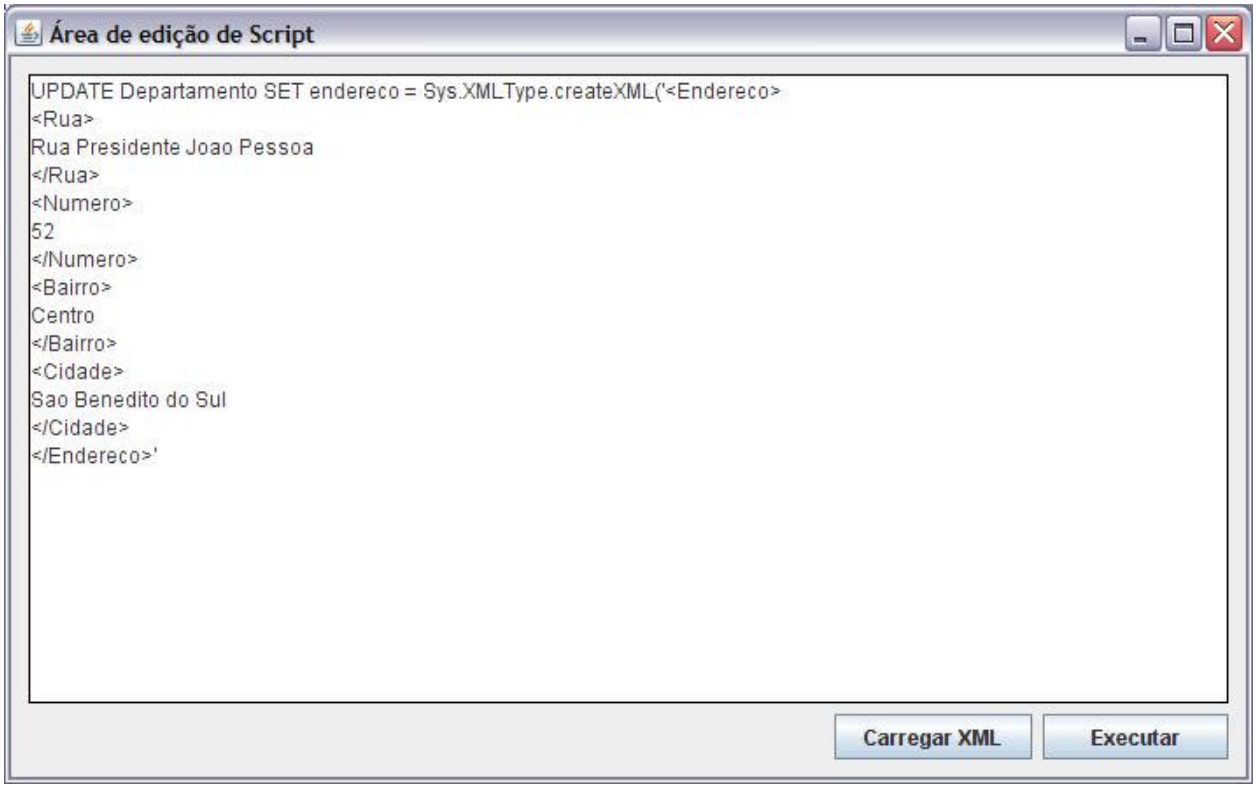

FIGURA 3.13 O SISTEMA CARREGA O ARQUIVO

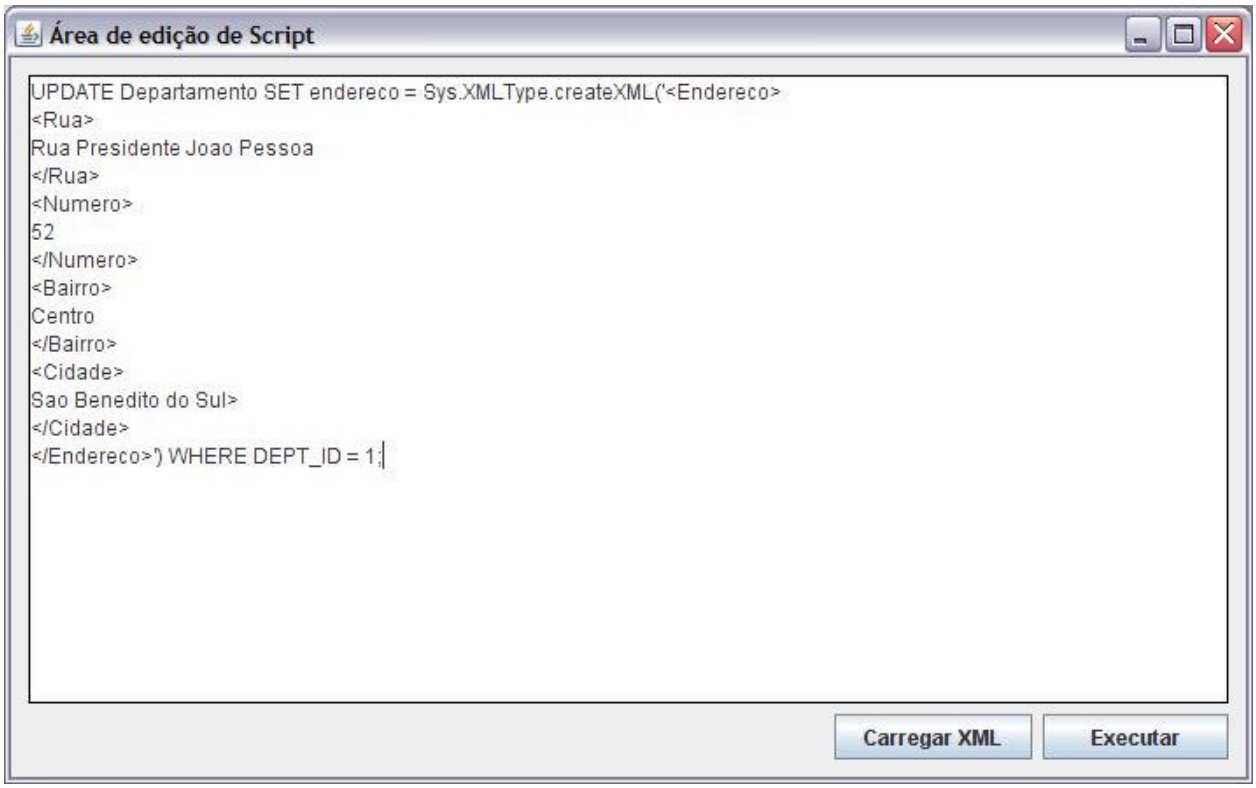

FIGURA 3.14 O USUÁRIO COMPLETA O SCRIPT

4 CONSIDERAÇÕES FINAIS

Desde o seu advento, XML tem se tornado o padrão mundialmente aceito para integração de banco de dados e transmissão de informações através da internet, além de ter possibilitado a revolução da chamada *web* semântica (AMIANO et al., 2006; BOURRET, 2008; MONTEIRO, 2008; SOUZA & ALVARENGA, 2004; STAKEN, 2001; SHANMUGASUNDARAM et al., 2003).

Tendo isso em vista, muitas tecnologias, entre elas sistemas de gerenciamento de bancos de dados com suporte a XML, foram criadas para o processamento do grande volume dos documentos, atendendo assim ao mercado. Todavia, há ainda a carência por um ambiente integrado de desenvolvimento, particularmente voltado para o Oracle XML DB.

Diante disto, o presente trabalho levantou, através de técnicas como a análise de ferramentas de desenvolvimento de bancos de dados correlatas, os requisitos de uma ferramenta capaz de suprir a carência descrita no parágrafo anterior. Assim, foram mostrados a modelagem e o desenvolvimento dos casos de uso [UC01], [UC02], [UC03], [UC04], [UC07], [UC09], [UC12], [UC13], [UC14] e [UC15] do XMLDB *Developer*. Além disso, procurou-se documentar uma pesquisa relacionando XML e sistemas de gerenciamento de banco de dados.

4.1 TRABALHOS FUTUROS

O XMLDB *Developer* foi pensado para ser desenvolvido através de uma metodologia evolutiva e incremental. Além disso, sua arquitetura também foi projetada tendo a escalabilidade como característica fundamental, de modo que o sistema possa ser expandido no futuro. Diante disto, seguem algumas sugestões de trabalhos futuros que podem agregar, substancialmente, valor ao sistema:

1. Integrar o XMLDB *Developer* ao Oracle SQL *Developer,* uma vez que o último é uma ferramenta madura que experimenta um crescimento em popularidade e tem uma série de funcionalidades que podem ser reaproveitadas e assim fazer do primeiro uma ferramenta mais interessante. Isto é possível porque o SQL *Developer* foi implementado através de uma arquitetura que permite a adição de novas funcionalidades por terceiros, através de uma API e da Oracle *Jdeveloper IDE*;
2. Submeter as funcionalidades implementadas a testes de usabilidade;
3. Elicitar novos requisitos para a ferramenta;
4. Criar uma infra-estrutura de colaboração comunitária para o projeto, como sítio, ferramenta de versionamento de código, fórum, lista de discussão, ferramenta de acompanhamento de defeitos, dentre outros, e assim tornar o projeto "*open-source*", e deste modo atrair a atenção da comunidade para que contribuam com a evolução do projeto.

Durante a fase de revisão da literatura, não foram encontradas referências a trabalhos similares, nos quais uma ferramenta com propósito bem definido de auxílio ao desenvolvimento e aprendizado do Oracle XMLDB fosse o principal objetivo.

REFERÊNCIAS BIBLIOGRÁFICAS

(AMIANO et al., 2006) AMIANO, M.; ETHIER, K.; D'CRUZ, C. ***XML - PROBLEM - DESIGN - SOLUTION***. John Wiley Consumer, 2006.

(APPLE, 2008) Apple. **Mac OS X Reference.** http://developer.apple.com/reference/MacOSX/index.html Último acesso em 25 de Junho de 2008.

(A-TEAM GROUP, 2008) A-TEAM Group. ***XML on Wall Street***. http://www.lighthouse-partners.com/xml/news_latest.htm Último acesso em 8 de Maio de 2008.

(BANERJEE et al., 2000) BANERJEE, S.; KRISHNAMURTY, V.; KRISHNAPRASAD, M.; MURHTY, R. **Oracle 8i - The XML Enabled Data Management System**. *ICDE2000, 2000.*

(BOURRET, 2008) BOURRET, R. ***XML and Databases***. http://www.rpbourret.com/xml/XMLAndDatabases.htm Último acesso em 30 de Março de 2008.

(EXIST, 2008) **eXist**. http://exist.sourceforge.net/ Último acesso em 30 de Março de 2008.

(GAMMA et al., 2000) GAMMA, E.; HELM, R.; JOHNSON, R.;VLISSIDES, J. **Padrões de projeto - Soluções Reutilizáveis de Software Orientado a Objetos.** Porto Alegre, Bookman, 2000.

(HAROLD & MEANS, 2004) HAROLD, E. R.; MEANS, W. S. ***XML IN A NUTSHELL***. OREILLY & ASSOC, 2004.

(HTML40, 2008) ***HTML 4.01 Specification***. http://www.w3.org/TR/html4/ Último acesso em 30 de Março de 2008.

(IBM, 2008) IBM. ***GML Set Reference***. http://publibfp.boulder.ibm.com/cgi-bin/bookmgr/BOOKS/dsm05m00/CCONTENTS Último acesso em 29 de Abril de 2008

(IBM, 2008) IBM. **Learning XML.** IBM Press Books.

(ISO/IEC 9075-14, 2003) ISO/IEC 9075-14. ***Information technology -- Database languages -- SQL -- Part 14: XML-Related Specifications (SQL/XML).*** 2003.

(KRISHNAPRASAD et al., 2005) KRISHNAPRASAD, M.; LIU, Z. H.; MANIKUTTY, A.; WARNER, J. W.; ARORA, V.; KOTSOVOLOS, S. **Query Rewrite for XML in Oracle XML DB**. Redwood Shores: Oracle Corporation, 2005.

(LIM & WEN, 2002) LIM, B.; WEN, H.. **The impact of next generation XML,** Information management & computer security, Volume 10 , Páginas 33-40, 2002.

(MICROSOFT, 2008) Microsoft. ***Access 2003 Documentation***. http://office.microsoft.com/en-us/access/FX100646921033.aspx Último acesso em 10 de Maio de 2008

(MICROSOFT, 2008) Microsoft. ***SQL Server 2005 Documentation***. http://www.microsoft.com/sql/techinfo/default.mspx Último acesso em 10 de Maio de 2008

(MICROSOFT, 2008) Microsoft. **Windows XP Documentation.** http://support.microsoft.com/kb/327405 Último acesso em 25 de Junho de 2008.

(MONTEIRO, 2008) MONTEIRO, J. ***História do XML***. http://www.criarweb.com/artigos/428.php Último acesso em 20 de Abril de 2008

(MYSQL, 2008) MySQL. ***MySQL Reference Manual.*** http://dev.mysql.com/doc/refman/4.1/en/index.html Último acesso em 14 de Abril de 2008

(ORACLE CORPORATION, 2007) Oracle Corporation. ***XML DB Home***. http://www.oracle.com/technology/tech/xml/xmldb/Current/xmldb_11g_twp.pdf Último acesso em 11 de Abril de 2008

(ORACLE, 2006) Oracle. ***Oracle SQL Developer*** http://www.oracle.com/technology/products/database/sql_developer/pdf/SQLDeveloper_whitepaper_v1.pdf Último acesso em 10 de Maio de 2008

(ORACLE, 2006) Oracle. ***Oracle XML DB Developer's Guide: Oracle 9iR2***. http://otn.oracle.com/tech/xml/xmldb Último acesso em 8 de Maio de 2008

(POSTGRESQL, 2008) PostgreSQL. ***PostgreSQL wiki***. http://wiki.postgresql.org/wiki/Main_Page Último acesso em 14 de Abril de 2008

(QUALITI, 2008) QUALITI. ***Qualiti Software Processes***. http://www.qualiti.com.br Último acesso em 28 de Maio de 2008

(RICCIO et al., 2008) RICCIO, E.; SAKATA, M.; MOREIRA, O.; QUONIAM, L. ***Introdução ao XBRL — nova linguagem para a divulgação de informações empresariais pela internet***. http://www.scielo.br/scielo.php?script=sci_arttext&pid=S0100-19652006000300016&lng=e&nrm=iso&tlng=e Último acesso em 8 de Maio de 2008

(SAX PROJECT, 2008) SAX Project. ***SAX Overview.*** http://www.saxproject.org/sax1-roadmap.html Último acesso em 15 de Abril de 2008

(SAXONICA, 2008) **Saxonica**. http://www.saxonica.com/documentation/index/intro.html Último acesso em 30 de Março de 2008

(SHANMUGASUNDARAM et al, 2003) SHANMUGASUNDARAM, J.; TUFTE, K.; HE, G.; ZHANG, C.; DEWITT, D.; NAUGHTON, J. ***Relational Databases for Querying XML Documents: Limitations and Opportunities***. http://www.cs.ubc.ca/~rap/teaching/504/2005/readings/relational-xml.pdf Último acesso em 10 de Março de 2008

(SOMMERVILLE, 2007) SOMMERVILLE, I.. ***Engenharia de Software***. Addison Wesley Bra, 2007.

(SOUZA & ALVARENGA, 2004) SOUZA, R. R.; ALVARENGA, L. (2004). ***A Web Semântica e suas contribuições para a ciência da informação***. http://www.scielo.br/pdf/ci/v33n1/v33n1a16.pdf Último acesso em 10 de Março de 2008

(STAKEN, 2001) STAKEN, K. ***www.xml.com***. Introduction to Native XML Databases: http://www.xml.com/pub/a/2001/10/31/nativexmldb.html Último acesso em 15 de Abril de 2008

(STEEGMANS et al., 2004) STEEGMANS, B.; BOURRET, R.; CLINE, O.; GUYENNET, O.; KULKARNI, S.; PRIESTLEY, S.; ***XML for DB2 Information Integration***. IBM RedBooks, 2004.

(SUN MICROSYSTEMS, 2008) Sun Microsystems. ***Java White Paper***. http://java.sun.com/white/ Último acesso em 10 de Maio de 2008

(SUN MICROSYSTEMS, 2008) Sun Microsystems. ***MySQL Documentation***. http://dev.mysql.com/doc/ Último acesso em 10 de Maio de 2008

(TLDP, 2008) TLDP. **The Linux Documentation Project**.. http://tldp.org/ Último acesso em 25 de Junho de 2008

(W3C, 2008) W3C. **SGML Resources**. http://www.w3.org/MarkUp/SGML/ Último acesso em 29 de Abril de 2008

(W3C, 2008) W3C., **Extensible Markup Language (XML) 1.0**:
http://www.renderx.com/~renderx/Demos/fo2html/xml.pdf Último acesso em 10 de Março de 2008

(W3C, 2008) W3C. ***DTD*** http://www.w3.org/TR/REC-html40/sgml/dtd.html Último acesso em 30 de Março de 2008

(W3C, 2008) W3C. ***Info set***. disponível em http://www.w3.org/TR/xml-infoset/ Último acesso em 15 de Abril de 2008

(W3C, 2008) W3C.. **Document Object Model Specification**. http://www.w3.org/TR/REC-DOM-Level-1/ Último acesso em 30 de Março de 2008

(W3C, 2008) W3C. ***XML Schema***. http://www.w3.org/XML/Schema Último acesso em 30 de Março de 2008

(W3C, 2008) W3C. ***XPath Specification***. http://www.w3.org/TR/xpath Último acesso em 30 de Março de 2008

(W3C, 2003) W3C. ***XQuery 1.0: An XML Query Language***. W3C Working Draft, 2003.

(W3C, 2008) W3C. ***XSLT Specification***. http://www.w3.org/TR/xslt Acesso em 30 de Março de 2008

(XBRL, 2008) XBRL. ***XBRL Specifications***. http://www.xbrl.org/Specifications/ Último acesso em 8 de Maio de 2008

(XMLDB INITIATIVE, 2008) XML:DB Initiative. http://xmldb-org.sourceforge.net/ Último acesso em 25 de Junho de 2008

Assinaturas

Prof. Ph.D. Fernando da Fonseca de Souza

Orientador

Yuri Cesar Serapião Soares Pereira

Aluno