NATHÁLIA DE ANDRADE TEIXEIRA

UTILIZAÇAO DE XML PARA IMPORTAÇÃO E EXPORTAÇÃO DE DADOS

Assis
2008

UTILIZAÇAO DE XML PARA IMPORTAÇÃO E EXPORTAÇÃO DE DADOS

NATHÁLIA DE ANDRADE TEIXEIRA

Trabalho de Conclusão de Curso apresentado ao Instituto Municipal de Ensino Superior de Assis, como requisito do Curso de Graduação, analisado pela seguinte comissão examinadora:

Orientador: Prof. Dr. Alex Sandro Romeo de Souza Poletto

Analisador (1): Luiz Ricardo Begosso

Analisador (2): Domingos de Carvalho Villela Júnior

Assis
2008

NATHÁLIA DE ANDRADE TEIXEIRA

UTILIZAÇAO DE XML PARA IMPORTAÇÃO E EXPORTAÇÃO DE DADOS

Trabalho de Conclusão de Curso apresentado ao
Instituto Municipal de Ensino Superior de Assis,
como requisito do Curso se Graduação.

Orientador: Prof. Dr. Alex Sandro Romeo de Souza Poletto

Área de Concentração: Sistemas de Bancos de Dados; XML; PL/SQL.

Assis
2008

DEDICATÓRIA

Dedico este trabalho aos meus pais Silvestre e Izabel Cristina, ao meu irmão Rafael e a todos estiveram ao meu lado.

AGRADECIMENTOS

Ao Prof. Dr. Alex Sandro Romeo de Souza Poletto pela orientação e pelo constante estímulo transmitido durante o trabalho.

A Prof. Marisa Atsuko Nitto pelo apoio e estímulo.

Aos meus amigos, Danilo Ribeiro, Danilo Coelho, Junior Mazalli (Bob), Edi, Luiz e Tatiani que apesar das dificuldades e limitações que também enfrentaram, sempre estiveram do meu lado nos momentos em que precisei.

Aos meus pais Silvestre e Izabel Cristina, ao meu irmão Rafael e minha prima Bel por acreditarem na minha capacidade, pela confiança e força transmitidas e pela compreensão nos momentos de dificuldade.

A Deus, Nossa Senhora e ao Espírito Santo que me iluminaram nas horas em que mais precisei.

RESUMO

A necessidade de se trocar informações de maneira segura, eficiente e padronizada, entre diferentes aplicações, levou a crescer o uso da Linguagem XML (eXtensible Markup Language – Linguagem de Marcação Extensível), já que a mesma é uma linguagem voltada para o gerenciamento de dados, além de ser multi-plataforma, e de garantir a integridade dos dados durante sua exportação e importação. Este trabalho apresenta a Linguagem XML, conceitos sobre Banco de Dados, bem como a forma como são realizados os processos de importação e exportação de dados.em formato XML.

Palavras-chave: Banco de Dados, XML, PL/SQL

ABSTRACT

The need to exchange information in a secure manner, efficient and standardized among different applications, has led to growing use of XML (eXtensible Markup Language - Extensible Markup Language) as it is a language facing the management data, in addition to being multi-platform, and ensure data integrity during their export and import. This paper presents the XML, Database concepts and the way they are made and the procedures for import and export data in format XML.

Keywords: Database, XML, PL/SQL.

LISTA DE ILUSTRAÇÕES

LISTA DE ABREVIATURAS E SIGLAS

XML	eXtensible Markup Language
W3C	World Wide Web Consortium
SGML	Standard Generalized Markup Language
DTD	Document Type Definition
XSD	XML Schema Definition
API	Application Programming Interfaces
XSLT	eXtensible Stylesheet Language Transformer
DOM	Document Objetc Model
SAX	Simple API for XML
SBD	Sistema de Banco de Dados
SGBD	Sistema Gerenciador de Banco de Dados
DBA	Database Administrator
DDL	Data Definition Language
DML	Data Manipulation Language
HTML	HyperText Markup Language
HTTP	Hypertext Transfer Protocol
SOAP	Simple Object Acess Protocol

SUMÁRIO

1. INTRODUÇÃO

A necessidade de se criar uma padronização segura e eficiente para importação e exportação de informações, entre aplicações de diversos fins, desenvolvidas com ferramentas diferentes, podendo os dados estarem armazenados em diferentes sistemas de bancos de dados, levou ao surgimento dos arquivos XML (eXtensible Markup Language – Linguagem de Marcação Extensível).

Atualmente, o meio mais largamente utilizado para a troca de informações entre aplicações são os arquivos de formato texto (.txt) O problema em se usar esse tipo de arquivo, em intercâmbios de dados, obriga os programas responsáveis pela exportação e importação a conhecer bem o formato utilizado no processo, caso contrário ocorrerão falhas, além disso, arquivos com extensão “.txt” podem ser facilmente violados, podendo prejudicar a integridade das informações.

Por razão desses problemas, surgiu então a utilização da XML, no sentido de oferecer um meio mais seguro, adaptável as exigências atuais, e padronizada para a troca de informações entre aplicações.

A XML é uma linguagem de marcação voltada para o gerenciamento de documentos, sendo uma linguagem padrão e adaptável, podendo até mesmo ser convertido para outros formatos. Esses motivos levam a XML a ser cada vez mais utilizada.

1.1 OBJETIVOS

Este trabalho tem por objetivo mostrar o porquê da utilização cada vez mais forte da XML para manipulação de informações, ilustrando na prática, como as exportações e importações são realizadas.

Além disso, como um segundo objetivo, tem-se a preocupação de apresentar conceitos e noções da XML, bem como toda a estrutura de um arquivo XML.

1.2 JUSTIFICATIVAS

Diante dos problemas encontrados na manipulação e trocas de dados em formato texto (.txt), surge então a necessidade de se utilizar um outro formato no sentido de contornar os tais problemas. É partir desta necessidade que a XML vem se tornando uma linguagem padrão em trocas de informações entre aplicações.

Algumas das características que justificam o uso da XML é que a mesma pode ser facilmente entendida por ser auto-documentável, é multi-plataforma, e pode ser convertida para outros formatos.

1.3 MOTIVAÇÃO

Para se tornarem fortes e sobreviverem aos imprevistos, às empresas necessitam de ferramentas que as ajudem e as auxiliem em suas tomadas de decisões, e um bom sistema de banco de dados bem modelado e construído, é uma das ferramentas mais importantes.

As informações contidas nesses sistemas de bancos de dados precisam ser compartilhadas entre as aplicações, para que assim todos os interessados estejam bem informados.

Para garantir o sucesso durante o processo de troca de dados entre as aplicações, surgi o uso da XML, que cada vez mais vem aumentando, podendo em um futuro próximo ser uma linguagem padrão para manipulação e troca de dados. Esta é a razão para a realização deste trabalho.

1.4 ESTRUTURA DO TRABALHO

Este trabalho será dividido em seis capítulos, sendo o primeiro esta introdução.

No segundo capítulo, serão apresentados os conceitos sobre a linguagem XML, sua origem e estrutura.

No terceiro capítulo, serão apresentados os objetivos do Sistema de Banco de Dados, conceitos e noções sobre Sistema Gerenciador de Banco de Dados, Administrador e Usuários de Banco de Dados, Banco de Dados Relacionais e chaves primárias e estrangeiras.

No quarto capítulo, serão apresentados os recursos para armazenar e trocar dados XML.

No quinto capítulo, será apresentada a proposta de trabalho e como deverá ser aplicada.

No sexto capítulo, será apresentado o estudo de caso e como foi desenvolvido.

No sétimo capítulo, serão apresentadas as considerações finais e as projeções futuras.

No oitavo capítulo serão citadas as referências bibliográficas utilizadas no desenvolvimento do trabalho.

2. XML

Neste capítulo, serão apresentados os conceitos sobre XML, bem como sua origem e estrutura.

2.1 ORIGEM DA XML

A XML foi desenvolvida em 1996 pela W3C (World Wide Web Consortium), com base na SGML (Standard Generalized Markup Language – Linguagem de Marcação Padrão Generalizada) porém, mais simples e eficaz, podendo representar dados, importá-los e exportá-los de uma aplicação para outra com mais facilidade.

2.2 ESTRUTURAS DE DADOS XML

A estrutura de um documento XML consiste em elementos, que são pares de *tags* que marcam o início e o fim do documento, contendo entre elas o corpo do texto.

Em documentos XML é permitido apenas o uso de um elemento raiz, chamado também de elemento pai, e entre o início e o fim desse elemento é onde se encontram os subelementos ou elementos filhos, que devem se aninhamos corretamente.

A XML não limita a quantidade de tags em um documento, elas são controladas pelo próprio desenvolvedor, podendo ele dar o nome que desejar para cada tag, e esses nomes podem ser repetidos em um mesmo documento.

A Figura 1 apresenta um exemplo de documento XML, onde <carta></carta> é o elemento pai, e as outras *tags* são os elementos filhos.

```
<carta>
      <cabeçalho>
         <remetente>Maria</remetente>
         <destinatário>João</destinatário>
      </cabeçalho>
     <parágrafo>Olá</parágrafo>
     <parágrafo>Tchau</parágrafo>
</carta>
```

Figura 1. Estrutura de Documento XML

O fato das *tags* poderem ser repetidas em um documento XML pode tornar defeituosa a representação desses dados, porém, quando se trata da troca desses dados entre aplicações, essas *tags* tornam o documento fácil de ser entendido, já que a mensagem se torna auto-documentável.

2.3 ESQUEMAS DA XML

Os esquemas servem para definir e restringir os tipos de dados que podem ser armazenados em um documento XML. Quando um documento XML segue as regras estabelecidas de um esquema pode-se dizer que ele é válido.

A seguir será descrita as duas linguagens de definição de esquema para XML, a DTD (Document Type Definition – Definição do Tipo do Documento) e a XMLSchema.

2.3.1 DTD

A DTD é uma linguagem que define as regras de como deve ser a estrutura de um documento XML. É por meio de uma DTD que é realizada uma análise do

documento XML dizendo se o mesmo está estruturado corretamente ou não. O uso da DTD é opcional.

A Figura 2 é um exemplo de uma DTD da Figura 1, em que <!DOCTYPE> é o tipo do documento, <!ELEMENT> são os tipos de elementos do documento e (#PCDATA) corresponde aos dados de caractere a ser inserido.

```
<!DOCTYPE CORRESPONDENCIA [
    <!ELEMENT carta (cabeçalho | parágrafo) >
    <!ELEMENT cabeçalho (remetente | destinatário) >
    <!ELEMENT parágrafo (#PCDATA) >
] >
```

Figura 2. DTD da Figura 1

2.3.2 XML Schema Definition

A XSD (XML Schema Definition) tem o mesmo papel que a DTD que é a de especificar os tipos de elementos em um documento XML e ordená-los, porém, a XSD possui algumas vantagens sobre a DTD como: a XSD pode especificar um elemento como sendo do tipo *integer, string, boolean, date, decimal*; permite que o usuário crie seus tipos definidos; permite o uso de chave estrangeira e recursos de exclusividade; utiliza de recursos de herança; permite utilizar valores mínimos e máximos para restringir os tipos de elementos.

Além dessas vantagens, a XSD é derivada da XML, o que permite que os recursos da XML possam ser utilizados dentro da linguagem XSD, ao contrário da DTD, que necessita de uma API (Application Programming Interfaces- Interfaces de Programas de Aplicações) para manipulá-la. Porém a XSD não pode para declarar entidades como a DTD.

A Figura 3 mostra um exemplo de um documento XSD.

```
<xs:schema xmlns=”http://www.w3.org/2001/XMLSchema>
<xs: element name = “carta”/>
        <xs: complexType>
        <xs: sequence>
        <xs: element name=”cabecalho” >
                <xs: element name=”remetente” type=”xs:string”/>
                <xs: element name=”destinatario” type=”xs:string”/>
        </xs: element>
        <xs: element name=”paragrafo” type=”xs:string”/>
        </xs: sequence>
        </xs: complexType>
</xs:schema>
```

Figura 3. XMLSchema da Figura 1

2.4 X PATH

XPath é uma linguagem utilizada para endereçar e encontrar partes de documentos XML. Esses endereços são construídos através de expressão de caminho onde são especificados os caminhos que devem ser percorridos para chegar até a parte do documento desejado. Para construir essas expressões são utilizados / para separar os caminhos.

A Figura 4 mostra um exemplo de consulta da Figura 1, utilizando XPath:

```
/carta/cabeçalho/remetente
```

Figura 4. Consulta XPath da Figura 1

2.5 XQUERY

XQuery é uma linguagem padrão de consulta para tipos de dados XML. Essa linguagem é formada por uma seqüência de comandos, são eles: *for, let, where*,

order by e *return*. Comandos esses mais conhecidos como a expressão FLWOR. Esses comandos foram baseados nos comandos da SQL, porém são diferentes.

A Figura 5 traz um exemplo de uma consulta realizada em XQuery.

```
for $x in /banco-2/conta
let $numconta: = $x/numero_conta
where $x/ saldo > 400
return <numero_conta> { $numconta} </numero_conta>
```

Figura 5. Consulta em XQuery (Silberschatz , 2006, p.274)

2.6 XSLT

A XSLT (eXtensible Stylesheet Language Transformer) é uma linguagem utilizada para transformar documentos XML em outros formatos como HTML, XHTML ou até mesmo em outros documentos XML.

A XSLT está embutida na XSL (eXtensible Style Language) que é a linguagem que gera a folha de estilo para XML, onde será declarado o tipo de formatação a ser utilizada nos documentos XML, assim como margens, fontes, etc.

No processo de transformação de documentos XML, a XSLT utilizada templates para selecionar os nós da árvore através de expressão da XPath.

A XSLT pode também ser utilizada para a realização de consultas.

2.7 XMLELEMENT ( )

A XMLELEMENT ( ) é uma função que gera e retorna elementos em formatado XML, através do fornecimento do nome da tabela e do campo que se deseja recuperar.

A seguir será apresentado um exemplo de XMLELEMENT (), onde serão informados o nome da tabela que no caso é “professor” e o campo “nome”:

```
select xmlelement("nome",nome)
as xml_customer
from professor;
```

Figura 6. XMLELEMENT ( )

O resultado traz todos os nomes da tabela professor:

XML_CUSTOMER
<nome>ALEX</nome>
<nome>ALMIR</nome>
<nome>BEGOSSO</nome>
<nome>GUTO</nome>
<nome>TALO</nome>
<nome>DOMINGOS</nome>
<nome>OSMAR</nome>
<nome>URSO</nome>
<nome>FABIO</nome>
<nome>DOUGLAS</nome>

Figura 7. Resultado XMLELEMENT ( )

2.8 XMLATTRIBUTES ( )

“XMLATTRIBUTES ( ) é usada em conjunto com XMLELEMENT ( ) para especificar os atributos dos elementos XML recuperados por XMLELEMENT ( )” (PRICE, 2008, p 636). XMLATTRIBUTES ( ) serve também para dar nomes aos atributos.

Segue exemplo de XMLATTRIBUTES ( ):

```
select xmlelement(
"professor",
xmlattributes(
codigo as "cod",
nome as "name",
to_char (nascimento, 'mm/dd/yyyy')as "nasc"
)
)
as xml_customers
from professor
where codigo in (1,2);
```

Figura 8. XMLATTRIBUTES( )

Resultado:

XML_CUSTOMERS
<professor cod="1" name="ALEX" nasc="11/17/1971"></professor>
<professor cod="2" name="ALMIR" nasc="05/01/1971"></professor>

Figura 9. Resultado XMLATTRIBUTES ( )

2.9 XMLFOREST ( )

A XMLFOREST ( ) é usada para gerar uma floresta de elementos XML. Ela também concatena os elementos sem precisar utilizar o operador ||.

Segue um exemplo de consulta utilizando XMLFOREST ( ):

```
select xmlelement(
"professor",
xmlforest(
codigo as "cod",
cargo as "carg",
to_char (nascimento, 'mm/dd/yyyy') as "nasc"
)
)
as xml_customers
from professor
where codigo in (1,3);
```

Figura 10. XMLFOREST ( )

Resultado da consulta:

XML_CUSTOMERS
<professor><cod>1</cod><carg>1</carg><nasc>11/17/1971</nasc></professor>
<professor><cod>3</cod><carg>3</carg><nasc>06/09/1971</nasc></professor>

Figura 11. Resultado XMLFOREST ( )

2.10 XMLAGG ( )

“Normalmente a XMLAGG ( ) é utilizada para agrupar código XML em uma lista de itens comuns abaixo de um único pai ou para recuperar dados de coleções” (PRICE, 2008, p 637).

O exemplo abaixo utiliza a cláusula order by em XMLAGG ( ):

```
select xmlelement(
"professor",
xmlagg(
xmlelement("professor", nome)
order by cargo asc
)
)
as xml_customers
from professor
where codigo in (1,2);
```

Figura 12. XMLAGG ( )

Resultado do select:

XML_CUSTOMERS
<professor><professor>ALEX</professor><professor>ALMIR</professor></professor>

Figura 13. Resultado XMLAGG ( )

2.11 XMLCOLATTVAL ( )

“Você usa XMLCOLATTVAL ( ) para criar um fragmento de código XML e depois expandir o código resultante. Cada fragmento tem a coluna de nome com o nome do atributo” (PRICE, 2008, p 640).

A cláusula As é utilizada para mudar o nome do atributo.

Exemplo utilizando XMLCOLATTVAL ( ) e a cláusula As:

```
select xmlelement(
"professor",
xmlcolattval(
codigo as "cod",
nome as "nome",
nascimento as "nasc"
)
)
as xml_customers
from professor
where codigo in (1,2);
```

Figura 14. XMLCOLATTVAL ( )

Resultado:

XML_CUSTOMERS

```
<professor>
<column name = "cod">1</column>
<column name = "nome">ALEX</column>
<column name = "nasc">1971-11-17</column>
</professor>
<professor>
<column name = "cod">2</column>
<column name = "nome">ALMIR</column>
<column name = "nasc">1971-05-01</column>
</professor>
```

Figura 15. Resultado XMLCOLATTVAL ( )

2.12 XMLCONCAT ( )

A XMLCONCAT ( ) serve para concatenar elementos.

Exemplo de XMLCONCAT ( ) utilizando a tabela professor:

```
select xmlconcat(
xmlelement("nome", nome),
xmlelement("nasc",
nascimento),
xmlelement("cargo", cargo)
)
as xml_customers
from professor
where codigo in (1,2);
```

Figura 16. XMLCONCAT ( )

Resultado:

XML_CUSTOMERS
<nome>ALEX</nome><nasc>1971-11-17</nasc><cargo>1</cargo>
<nome>ALMIR</nome><nasc>1971-05-01</nasc><cargo>2</cargo>

Figura 17. Resultado XMLCONCAT ( )

2.13 XMLPARSE ( )

“XMLPARSE ( ) é usada para analisar e gerar código XML a partir do resultado avaliado de uma expressão” (PRICE, 2008, p 641).

Exemplo utizando XMLPARSE ( ) na tabela professor:

```
select xmlparse(
content
'<professor><codigo>1</codigo><nome>Alex</nome></professor>'
wellformed
)
as xml_customers
from dual;
```

Figura 18. XMLPARSE ( )

Resultado:

XML_CUSTOMERS
<professor><codigo>1</codigo><nome>Alex</nome></professor>

Figura 19. Resultado XMLPARSE ( )

2.14 XMLPI ( )

“XMLPI ( ) gera uma instrução de processamento XML. Normalmente, uma instrução de processamento é usada para fornecer a um aplicativo informações associada a dados XML; o aplicativo pode então usar a instrução de processamento para determinar como vai processar os dados XML” (PRICE, 2008, p 642).

Segue exemplo de XMLPI ( ):

```
select xmlpi(
name "order_status",
'placed,pending,shipped'
)
as xml_order_status_pi
from dual;
```

Figura 20. XMLPI ( )

Resultado:

XML_ORDER_STATUS_PI
<?order_status placed,pending,shipped?>

Figura 21. Resultado XMLPI ( )

2.15 XMLCOMMENT ( )

A XMLCOMMENT ( ) serve para inserir comentários em XML, através de string. O exemplo abaixo insere um comentário:

```
select xmlcomment(
'Exemplo de comentario XML'
)
as xml_comment
from dual;
```

Figura 22. XMLCOMMENT ( )

Resultado:

XML_COMMENT
<!--Exemplo de comentario XML-->

Figura 23. Resultado XMLCOMMENT ( )

2.16 XMLSEQUENCE ( )

“XMLSEQUENCE ( ) gera um objeto XMLSequenceType, que é um varray de objetos XMLType” (PRICE, 2008, p 643).

Segue exemplo de XMLSEQUENCE ( ):

```
select                                                    value
(list_of_values).getstringval()order_values
from table(
xmlsequence(
extract(
xmltype('<a><b>placed</b><b>pending<b/></a>'),
'/a/b'
)
)
)
list_of_values;
```

Figura 24. XMLSEQUENCE ( )

Resultado:

ORDER_VALUES
placed
pending

Figura 25. Resultado XMLSEQUENCE ( )

2.17 XMLSERIALIZE ( )

“XMLSERIALIZE ( ) gera uma representação de string ou ob de dados XML a partir do resultado avaliado de uma expressão” (PRICE, 2008, p 644).

```
select xmlserialize(
content
xmltype('order_status>shipped</order_status>
')
)
as xml_order_status
from dual;
```

Figura 26. XMLSERIALIZE ( )

Resultado:

XML_ORDER_STATUS
<order_status>shipped</order_status>

Figura 27. Resultado XMLSERIALIZE ( )

2.18 INTERFACES DE APLICAÇÕES PARA XML

Nesta sessão serão apresentadas as duas interfaces de aplicações para XML mais utilizadas: DOM (Document Objetc Model – Modelo de Objetos para Documento) e SAX (Simple API for XML - API Simples para XML).

2.18.1 DOM

O DOM é uma API (Application Programming Interface – Interface de Programação de Aplicações) utilizada para manipular documentos XML.

“DOM pode ser usado para acessar dados XML armazenados em banco de dados, e um banco e dados XML pode ser embutido com DOM como sua interface principal para acessar e modificar dados.Porém, a interface DOM não admite qualquer forma de consulta declarativa” (SILBERSCHATZ, 2006, p 279).

2.18.2 SAX

SAX é uma API baseada na manipulação de eventos. Quando uma leitura está sendo feita em um documento XML, ela gera dois eventos, um para mostrar o início da tag e outro para mostrar o fim da tag.

“SAX geralmente exige mais esforço de programação do que DOM, mas ajuda a evitar o trabalho adicional de criar uma árvore DOM em situações em que a aplicação precisa criar sua própria representação de dados” (SILBERSCHATZ, 2006, p 280).

3. SISTEMAS DE BANCO DE DADOS

Neste capítulo serão apresentados os objetivos de um SBD (Sistema de Banco de Dados), bem como conceitos e noções sobre Sistema Gerenciador de Banco de Dados, Linguagem de Definição de Dados, Administrador e Usuários de Banco de Dados, Bancos de Dados Relacionais, e de chaves primárias e estrangeiras.

3.1 OBJETIVOS DOS SISTEMAS DE BANCO DE DADOS

Banco de Dados é um conjunto de registros ou dados inter-relacionados e armazenados, que representam informações sobre algo específico (SILBERSCHATZ, 1993).

Um Banco de Dados tem um papel muito importante dentro das empresas. Muitas das decisões são tomadas com base nas informações geradas através do tratamento dos dados armazenados nos Bancos de Dados.

Dentre os objetivos dos Sistemas de Banco de Dados, destaca-se o de oferecer mecanismos, tais como aplicativos para a manipulação dos dados, assim como sua segurança.

Segundo Silberschatz (1993), para que um sistema de banco de dados seja eficiente, é necessário tomar alguns cuidados em sua criação, tais como:

Redundância e Inconsistência de Dados: a redundância e inconsistência de dados podem ocorrer quando um programa é desenvolvido por vários programadores em diversas linguagens, assim corre-se o risco de debater com dados idênticos em diversos arquivos.

Dificuldade no acesso aos dados: a dificuldade no acesso aos dados pode ocorrer quando um programa não foi desenvolvido para atender qualquer tipo de necessidade do usuário. Por exemplo, o chefe pode necessitar de um determinado relatório e o programa não oferece esse tipo de serviço.

Isolamento de dados: pode ocorrer quando os dados estão espalhados em diversos arquivos, tornando difícil a recuperação dos mesmos por meio de outro programa.

Anomalias de acesso concorrente: ocorre quando um sistema pode se acessado por mais de um usuário ao mesmo tempo, e com isso a sua atualização pode acarretar falhas.

Problemas de segurança: os usuários do banco de dados geralmente não precisam ter acesso a todas as informações do banco, assim o acesso deve ser restrito, permitindo que cada usuário acesse apenas aos dados que realmente necessite.

Problemas de integridade: “Os valores dos dados armazenados nos bancos de dados precisam satisfazer certos tipos de restrições de consistência” (SILBERSCHATZ et al.,1993, p.4).

Tais restrições estão armazenadas nos programas, e para que novas alterações possam ser realizadas por todos os programadores, é necessário que sejam feitas funções permitindo esse acesso.

3.2 SISTEMA GERENCIADOR DE BANCO DE DADOS

O Sistema Gerenciador de Banco de Dados (SGBD) é um conjunto de sistemas que permite o acesso as informações armazenadas no banco de dados, sendo assim possível recuperá-las e manipulá-las. Um bom SGBD deve oferecer:

Garantia de Integridade: os meios de definir as restrições de consistência dos dados devem ser fornecidos para o Administrador de Banco de Dados (DBA) pelo SGBD.

Restrição de acesso não autorizado: o SGBD deve restringir o acesso indevido aos dados, fazendo com que cada usuário tenha acesso apenas aos dados que realmente necessite.

Recuperação e Backup: o SGBD tem que garantir a recuperação de dados caso aconteça alguma falha, como queda de energia, erros de sistema e quebras de disco.

Controle de Concorrência: o SGBD deve assegurar que quando um banco de dados é acessado por mais de um usuário ao mesmo tempo, que seja feita apenas uma atualização de cada vez, para que não ocorra inconsistência de dados.

Múltiplas Interfaces: o SGBD deve fornecer diversas ferramentas de acesso ao banco de dados, assim como, interface de consulta para usuários casuais, linguagem de programação para programadores, etc.

Representação de Relações Complexas entre os Dados: o SGBD deve fornecer uma variedade de relacionamentos complexos entre os dados, assim como recuperar e atualizar dados relacionados.

3.3 LINGUAGEM DE DEFINIÇÃO DE DADOS

A Linguagem de Definição de Dados (*Data Definition Language* – DDL) é a responsável pela definição do esquema a ser utilizado pelo banco de dados. As tabelas geradas pela compilação dos comandos DDL são armazenados em um dicionário, ou o mesmo que metadados (SILBERSCHATZ, 1993).

“A estrutura da armazenagem e os métodos de acesso usados em um sistema de bancos de dados são especificados por um conjunto de definições em um tipo especial de DDL chamado Linguagem de Definição de Dados. O resultado da compilação destas definições é um conjunto de instruções para especificar a implementação de detalhes do esquema de banco de dados que estão normalmente escondidos dos usuários” .(SILBERSCHATZ et al., 1993, p.14)

3.4 LINGUAGEM DE MANIPULAÇÃO DE DADOS

“A Linguagem de Manipulação de Dados (Data Manipulation Language – DML) é a linguagem que permite aos usuários fazer o acesso ou manipular dados como organizado pelo modelo de dados apropriado” (SILBERSCHATZ et al.,1993, p.15).

A linguagem de manipulação de dados é dividida em dois tipos. *DML’s procedurais:* é a linguagem que requer do usuário qual dado é necessário e como obtê-lo*, e*

DML's não procedurais: é a linguagem que requer do usuário qual dado é necessário sem precisar especificar como obtê-lo.

3.5 ADMINISTRADOR DE BANCO DE DADOS

O Administrador de Banco de Dados ou Database Administrator (DBA) é o responsável pelo controle central dos dados e dos programas de acesso a eles.

Conforme Silberschatz (1993), as funções de um DBA são:

- Decidir qual a estrutura de armazenagem e a estratégia de acesso;
- Decidir quais as modificações de esquema e de organização física;
- Ser o elo de ligação com os usuários;
- Decidir as verificações de autorização para acesso aos dados;
- Decidir as especificações de restrição de integridade.

3.6 USUÁRIOS DE BANCO DE DADOS

Segundo Silberschatz (1993), existem quatro tipos diferentes de usuários de banco de dados:

Programadores de aplicativos: são programadores que interagem com o banco de dados através da DML.

Usuários de alto nível / sofisticados: usuários sofisticados interagem com o sistema sem escrever programas. Em vez disso, eles formulam suas demandas em uma linguagem de consulta. Cada consulta é submetida a um processador de consulta, cuja função é pegar um comando de DML e dividí-lo em instruções que o gerenciador do banco de dados compreenda

Usuários especializados*:* usuários que escrevem programas que não se ajustam aos padrões tradicionais de processamento de dados. Sistemas de projeto apoiado por

computador, sistemas especialistas, sistemas que armazenam dados com tipos complexos e sistemas modeladores de ambiente são alguns tipo desses programas.

Usuários ingênuos: são usuários que interagem com o banco de dados através dos programas aplicativos permanentes.

3.7 BANCOS DE DADOS RELACIONAIS

Em um Banco de Dados Relacional os dados são organizados por meio de tabelas, também conhecidas por relações. Essas relações são compostas de linhas (tuplas) e colunas (atributos), onde cada linha corresponde a um relacionamento entre o conjunto de valores.

3.7.1 Conceito de Chaves

O conceito de chaves em um banco de dados relacional é de grande importância, pois é por meio das chaves que é possível identificar entidades e estabelecer relacionamentos na tabela.

É por intermédio da chave primária que os dados da tabela são organizados, sendo também o principal meio de acesso a tabela. A chave estrangeira permite o relacionamento em uma tabela, e seus valores devem pertencer a chave primária.

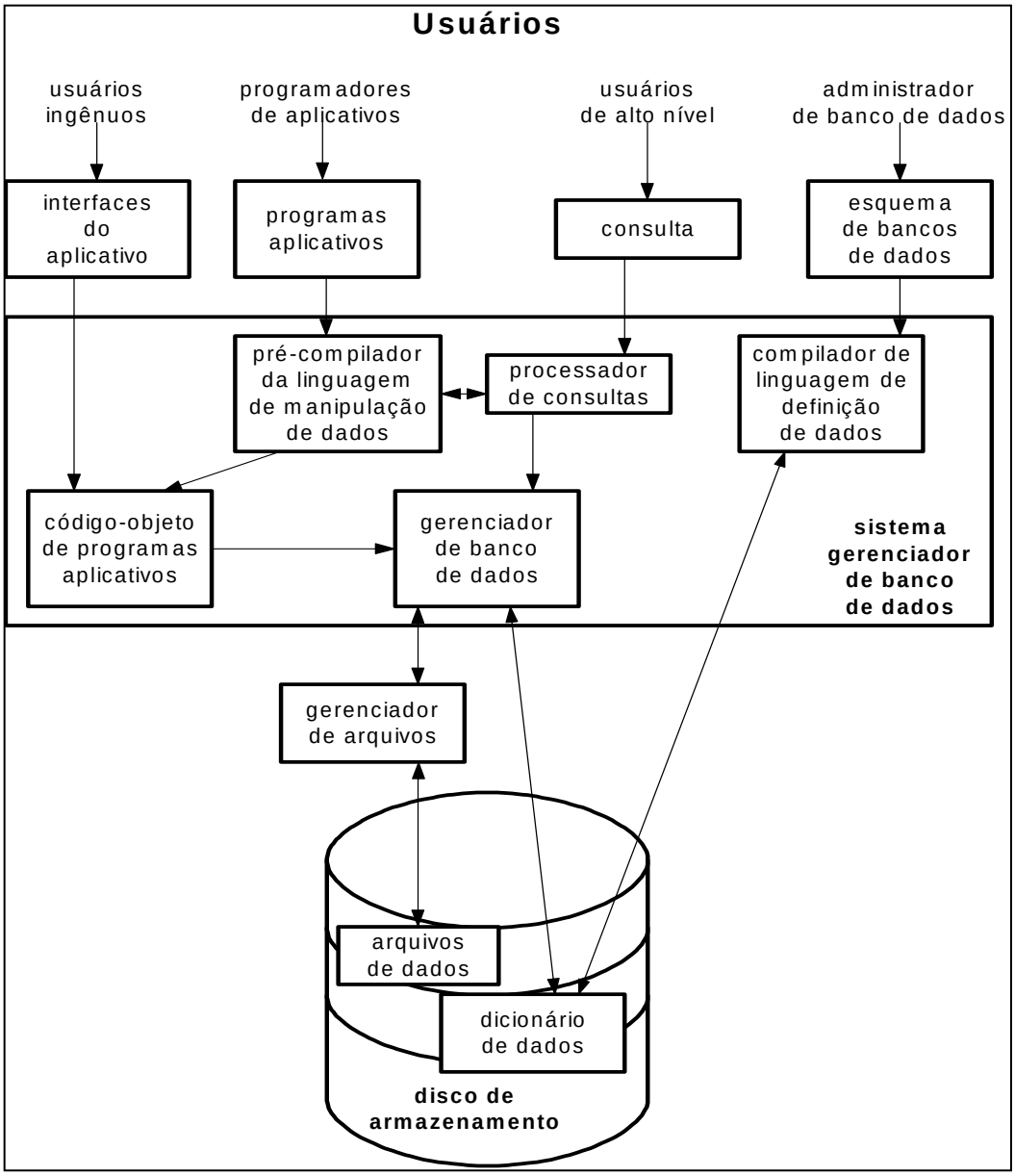

Figura 28. Sistema de Banco de Dados (Silberschatz et al.,1993, p.21)

4. MANIPULANDO DADOS XML

Neste capítulo serão apresentados recursos para armazenar e trocar dados XML.

4.1 ARMAZENANDO DADOS XML

Nesta seção serão apresentadas algumas maneiras de se armazenar dados XML.

4.1.1 Banco de Dados não Relacionais

Existem duas maneiras de se armazenar dados XML em um banco de dados não relacional:

Armazenar em arquivos simples: Embora essa técnica apresente diversas desvantagens, como não possuir acesso corrente, segurança e isolamento de dados, em algumas aplicações ela se torna suficiente, devido ao grande número de ferramentas de XML que trabalham com esse tipo de arquivos, sendo assim possível acessá-los e consultá-los.

Criar um banco de dados XML: Segundo Silberschatz (2006), os bancos de dados XML são bancos de dados que utilizam XML como seu modelo de dados básicos. Os primeiros banco de dados XML implementam o Document Object Model em um banco de dados orientado a objeto baseado em C++. Isso permite que grande parte da infra-estrutura do banco de dados orientado a objeto seja reutilizada, enquanto oferece uma interface XML padrão. O acréscimo de XQuery ou outras linguagens de consulta XML oferece consulta declarativa.

4.1.2 Banco de Dados Relacionais

Existem cinco maneiras de se armazenar dados XML em um banco de dados relacional:

Armazenar como string: pode-se armazenar pequenos documentos XML como string em tuplas no banco de dados, porém, apesar de simples de se usar, o banco de dados não reconhece o esquema dos elementos, tornando impossível as consultas, por mais simples que sejam.

Segundo Silberschatz (2006), uma das maneiras de contornar esse problema seria armazenar diferentes tipos de elementos em relações diferentes e armazenar os valores de alguns elementos críticos como atributos para ativar a indexação.

O banco de dados Oracle permite índices de função, fazendo com que não ocorra replicação de atributos entre a string XML e os atributos de relação.

Representação de árvore: qualquer tipo de dado XML pode ser armazenado como árvore utilizando um par de relações. Esse tipo de armazenamento permite que todas as informações XML possam ser representadas em um banco de dados relacional e também permite que consultas XML sejam realizadas dentro do banco de dados. A desvantagem em se usar esse tipo de armazenamento é que cada elemento é desmembrado em várias partes, o que exige que sejam feitas muitas associações para que os subelementos voltem para o elemento de origem.

Mapa de relações: Segundo Silberschatz (2006), os elementos XML cujo esquema é conhecido são mapeados para relações e atributos. Os elementos cujo esquema é desconhecido são armazenados como strings ou como uma árvore. Uma relação é criada para cada tipo de elemento (incluindo subelementos) cujo esquema é conhecido e cujo tipo é um tipo complexo (ou seja, contém atributos ou subelementos).

Publicando e fragmentando dados XML: “Quando a XML é usada para trocar dados entre aplicações corporativas, os dados normalmente são originados nos bancos de dados relacionais. Os dados nos bandos de dados relacionais precisam se publicados, ou seja, convertidos para o formato XML, para exportar para outras aplicações. Os dados que chegam precisam ser fragmentados, ou seja, convertidos de volta, de XML para o formato de relação normalizada, e armazenados em um Banco de Dados Relacional. Embora o código de aplicação possa realizar as operações de publicação e fragmentação, as operações são tão comuns que as

conversões devem ser feitas automaticamente, sem escrever o código da aplicação" (SILBERSCHATZ et al.,2006, p.282).

Armazenamento nativo dentro de um banco de dados relacional: Alguns sistemas de banco de dados estão suportando o armazenamento nativo de dados XML, assim, não há necessidade de transformar os dados para o formato relacional, já que eles são armazenados como strings ou em representação binária.

4.2 APLICAÇÕES XML

Nesta sessão, serão apresentadas algumas maneiras de armazenar, trocar dados XML.

4.2.1 Armazenando Dados com Estrutura Complexa

Esse tipo de armazenamento é utilizado quando há necessidade de trocar dados entre diferentes partes de uma aplicação.

"Por exemplo, um sistema de banco de dados pode representar um plano de execução de consulta (uma expressão de álgebra relacional com informações extras sobre como executar operações) utilizando XML. Isso permite que uma parte do sistema gere o plano de execução da consulta e outra parte o apresente, sem usar uma estrutura de dados compartilhada" (SILBERSCHATZ et al.,2006, p.284).

4.2.2 Formatos Padronizados para Troca de Dados

Aplicações específicas sobre diversas áreas como na de negócios, científicas, química entre outros, tem sido desenvolvidos com o padrão XML para importação e exportação de dados.

O uso de aplicações especializadas com padrão XML evita a redundância de dados, evitando a confusão entre atributos, o que ocorreria se a aplicação seguisse o modelo relacional.

4.2.2.1 Web Services

Muitas vezes as aplicações necessitam de dados armazenados em diferentes bancos de dados da organização ou de um determinado departamento. A organização não pode permitir o acesso direto ao banco de dados, então ela disponibiliza esses dados utilizando formatos baseados na Web, onde o usuário possa manipular esses dados retornando informações em formato HTML (HyperText Markup Language). Porém, quando esses dados precisam ser acessados por softwares, a organização utiliza do formato XML para especificar os valores de entrada e saída das consultas, utilizando o protocolo HTTP (Hypertext Transfer Protocol). A SOAP (Simple Object Acess Protocol) é responsável pela definição do padrão para que se possa usar a XML para representar entradas e saídas de procedimentos.

4.2.2.2 Mediação de Dados

A mediação de dados pode ser utilizada, por exemplo, quando se deseja obter uma relação de preços de um determinado item em diferentes sites. Quando esses sites oferecem esse tipo de informação em formato XML como um Web Service, é fácil extrair essa relação.

Os sites podem utilizar diferentes esquemas para representar as mesmas informações, o que fará com que a aplicação mediadora decida qual a melhor maneira de extrair e representar esses dados, isso geralmente exige muita transformação dos dados XML fazendo com que a XSLT e XQuery desempenhem seu papel.

5 PROPOSTA DE TRABALHO

Este trabalho tem por objetivo mostrar de que forma é realizada a importação e exportação de dados XML em um Banco de Dados Relacional.

Para demonstrar como é feito o intercâmbio de dados XML entre aplicações, será criada uma tabela com dados armazenados no formato XML, para que então, esses dados sejam exportados para outra tabela, para outro banco de dados, conforme ilustrado na Figura 29.

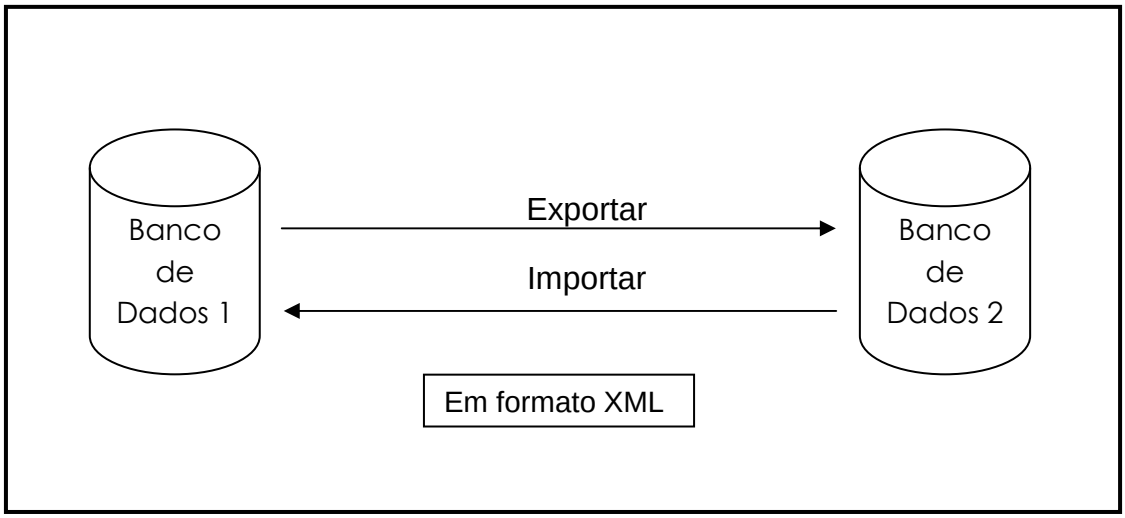

Figura 29. Proposta de Trabalho

5.1 FASES DA PROPOSTA

Para que possa ser realizado o intercâmbio de dados XML entre as tabelas, duas fases são necessárias, conforme apresentado a seguir.

5.1.1 Fase 1

Na primeira fase desta proposta, deve ser feito o levantamento do modelo de dados, ou seja, é feito um estudo sobre a estrutura que foi utilizada na construção da tabela, escolhida para exportar os dados, para que o processo de exportação possa ser realizado.

5.1.2 Fase 2

Na segunda fase da proposta serão preparados os arquivos XML, para que possam ser exportados para uma outra tabela por meio da especificação da expressão do seu caminho, conforme apresentado na Figura 30.

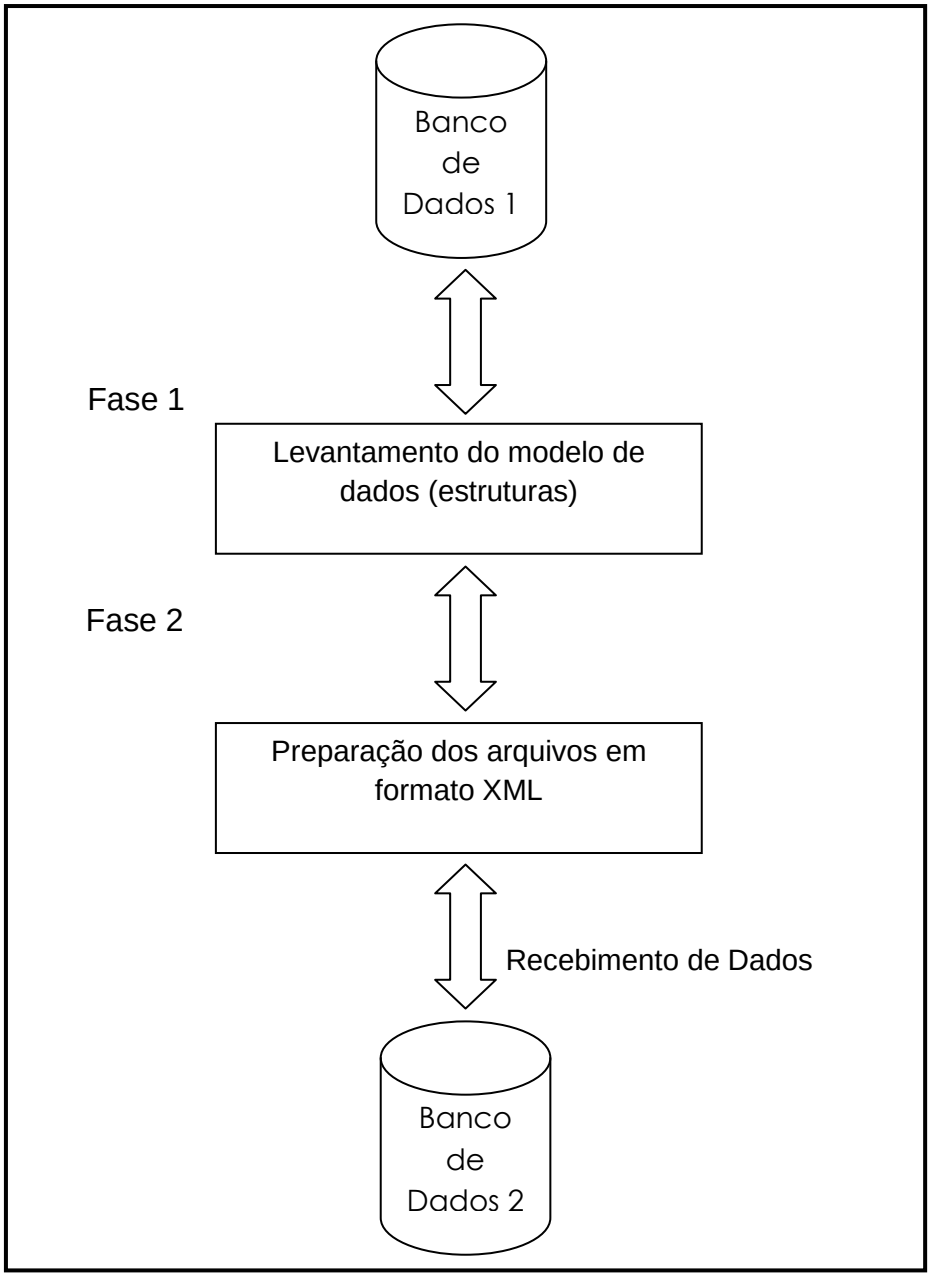

Figura 30. Fases da Proposta de Trabalho

6 ESTUDO DE CASO

O estudo de caso deste trabalho será desenvolvido por meio da demonstração de exportação e importação de dados de uma tabela através do Oracle 10g Express Edition e do Oracle 10g Standart.

A tabela utilizada neste estudo de caso é uma tabela cujo nome é cidades que armazena os campos código, nome, estado e qtd_habitantes (quantidade de habitantes). Os dados armazenados na tabela cidades, serão exportados para um documento XML, e em seguida serão importados para uma outra tabela nomeada cidades_xml.

6.1 EXPORTANDO DADOS XML DA TABELA CIDADES

Nesta sessão serão apresentados os processos de exportação no Oracle 10g Express Edition e no Oracle 10g Standart.

6.1.1 Exportando Dados XML no Oracle 10g Express Edition

A seguir serão demonstrados os passos necessários para que possa ser realizada a exportação de dados XML.

Ao acessar a *homepage* de banco de dados aparecerão os campos de *login* no banco de dados como ilustrado na Figura 31.

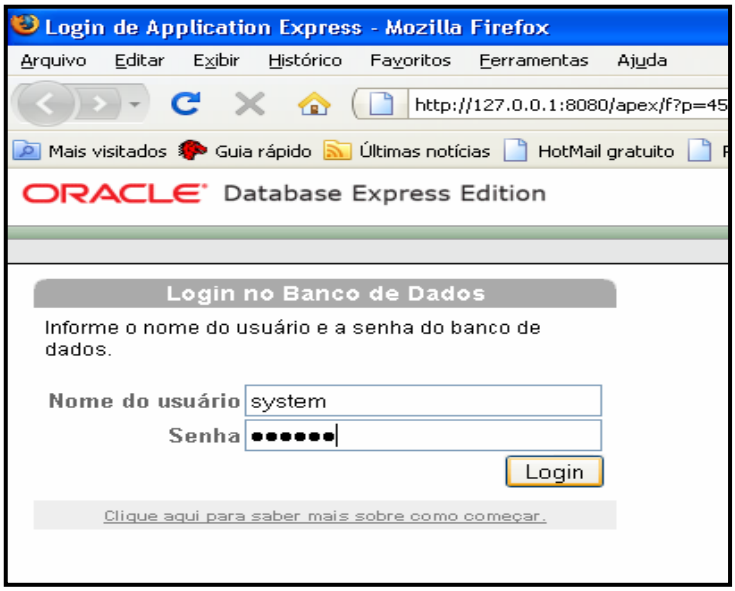

Figura 31. Login no Banco de Dados

A tela de início do banco de dados será exibida, e através dela, será feito o acesso aos Utilitários:

Figura 32. Utilitários do Banco de Dados

Ao abrir os Utilitários do banco de dados, será exibida a opção de Carga/Descarga de Dados, através dessa opção que é possível realizar a exportação e importação de dados XML.

Figura 33. Carga/Descarga de Dados no Banco de Dados

Ao clicar em Carga/Descarga de Dados, aparecerão três opções, das quais será escolhida a opção Descarregar.

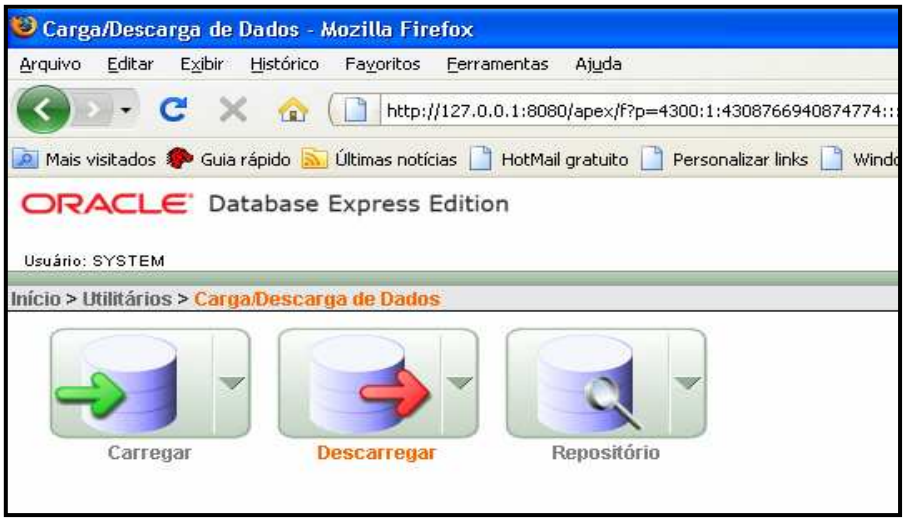

Figura 34. Descarregar Dados do Banco de Dados

A opção Descarregar oferece as possibilidade de exportar dados em formato texto e em formato XML.

Figura 35. Descarregar Dados XML do Banco de Dados

Nessa etapa será feita a conecção com o esquema do banco e dados onde está armazenada a tabela “cidades”, a ser exportada, em que o esquema é *system*.

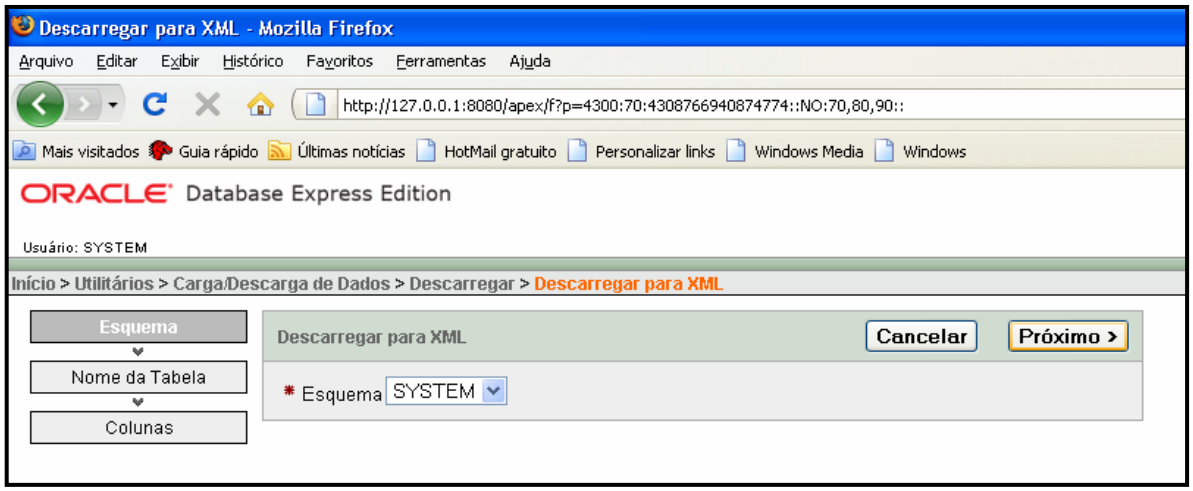

Figura 36. Conectar ao Esquema

Ao selecionar o esquema, e acessar a próxima etapa, serão exibidas todas as tabelas do banco de dados, onde se encontra a tabela “cidades”.

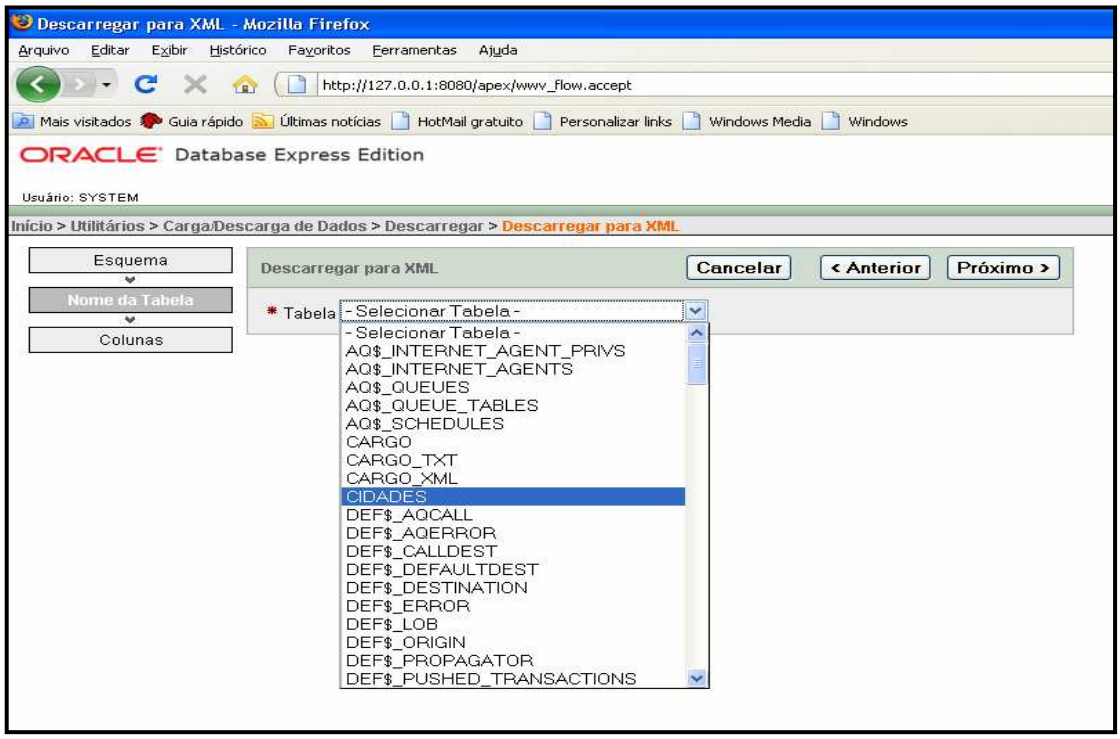

Figura 37. Selecionando a Tabela

Na etapa seguinte, podem-se escolher os campos da tabela a serem exportados, no caso, todos os campos da tabela “cidades” serão selecionados e a opção Exportar Como Arquivo será marcada.

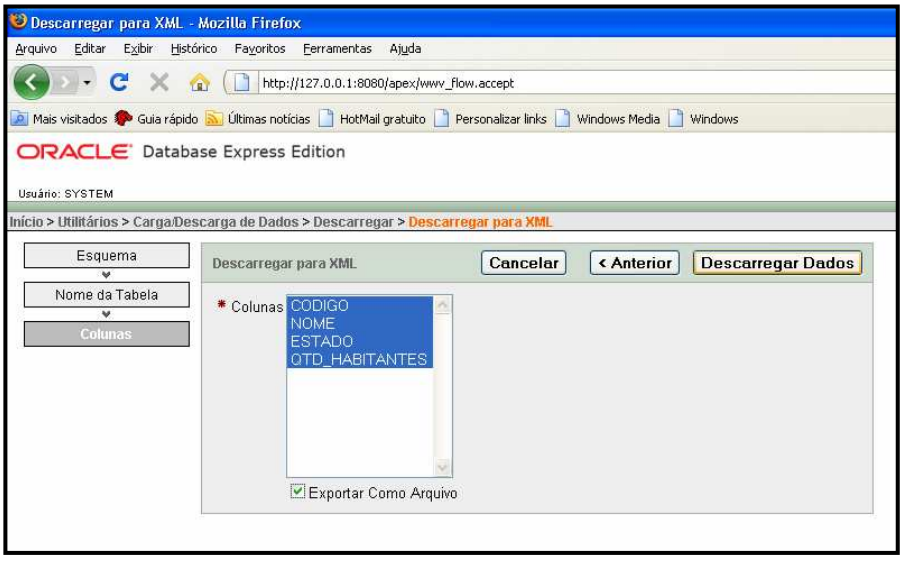

Figura 38. Selecionando Campos da Tabela

Após o clique no botão Descarregar Dados, aparecerá uma janela para indicar o diretório no qual será salvo o arquivo XML da tabela “cidades”.

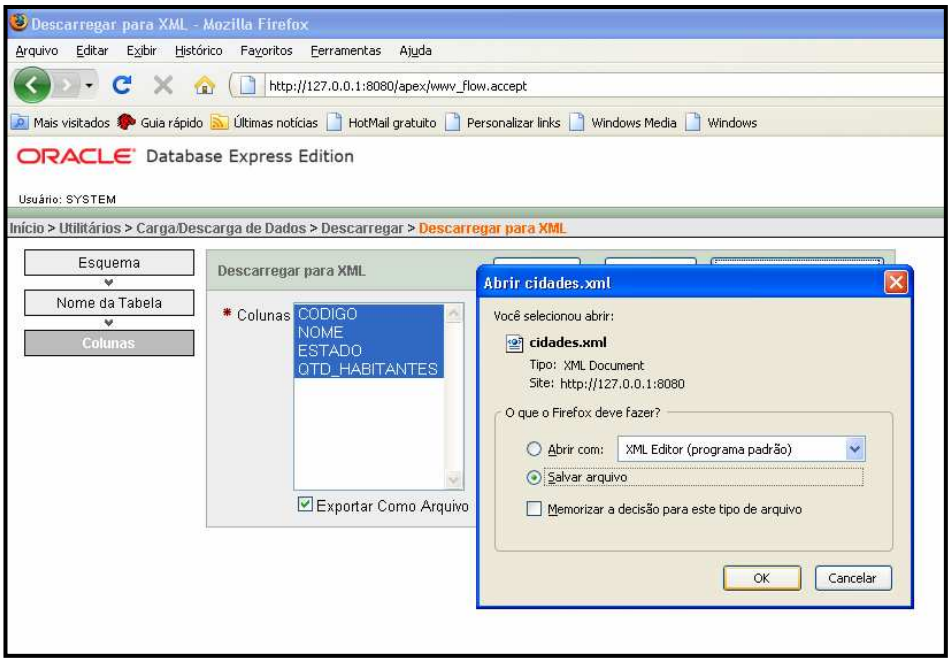

Figura 39. Escolher Diretório

O diretório escolhido foi o desktop, onde o arquivo XML da tabela “cidades” foi exportado. Ao abrir o arquivo XML, pode-se observar que os dados da tabela estão aninhados corretamente, e nomes dos campos da tabela se transformaram em *tags*, fazendo com que o documento se torne-ne auto-documentável e fácil de ser entendido.

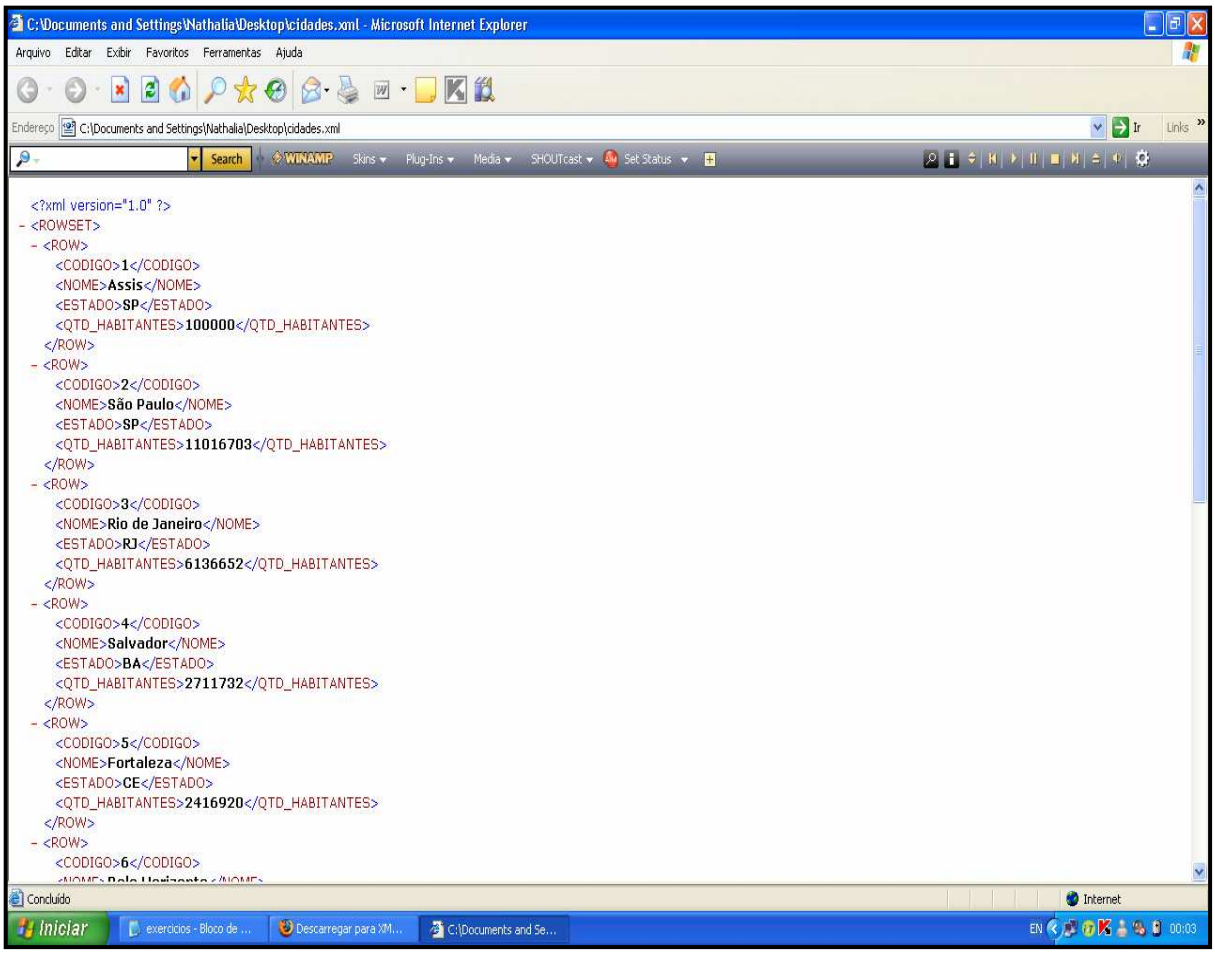

Figura 40. Arquivo XML Exportado da Tabela Cidades

6.1.2 Exportando Dados XML no Oracle 10g Standart

Para exportar dados XML de uma tabela no Oracle 10g Standart é necessário criar um diretório onde será salvo o documento XML.

```
CREATE DIRECTORY TEMP_FILES_DIR AS ‘C:\temp_files’;
```

Figura 41. Criando Diretório

Após criar o diretório, é preciso executar uma procedure onde será realizada a leitura da tabela cidades através de uma select.

A procedure ilustrada a seguir recupera os nomes da tabela cidades transformando em formato XML.

```
CREATE OR REPLACE PROCEDURE write_xml_data_to_file
(p_directory VARCHAR2,
p_file_name VARCHAR2
)
AS
v_file UTL_FILE.FILE_TYPE;
v_mount INTEGER := 32767;
v_xml_data XMLType;
v_char_buffer VARCHAR2(32767);
BEGIN
 v_file := UTL_FILE.FOPEN(p_directory,p_file_name,'w',v_mount);
 UTL_FILE.PUT_LINE(v_file,'<?xml version="1.0"?>');
 SELECT EXTRACT(
    XMLELEMENT("list_cidade",
      XMLAGG(
        XMLELEMENT("NOME",nome))),'/list_cidade'
 )
 AS xml_customers
 INTO v_xml_data
 FROM cidades;
 v_char_buffer := v_xml_data.GETSTRINGVAL();
 UTL_FILE.PUT(v_file,v_char_buffer);
 UTL_FILE.FFLUSH(v_file);
 UTL_FILE.FCLOSE(v_file);
END write_xml_data_to_file;
/
```

Figura 42. Procedure para Exportação XML

Criada a procedure é necessário executar a instrução *call,* onde será enviado o documento XML para o diretório criado.

```
CALL write_xml_data_to_file('TEMP_FILES_DIR','cidades.xml');
```

Figura 43. Chamando a Procedure

Ao acessar o diretório e abrir o documento XML, o seguinte resultado será exibido:

```
<?xml version="1.0" ?>

<ROWSET>
        <ROW>
                <NOME>Assis</NOME>
        </ROW>
        <ROW>
                <NOME>São Paulo</NOME>
        </ROW>
        <ROW>
                <NOME>Rio de Janeiro</NOME>
        </ROW>
        <ROW>
                <NOME>Salvador</NOME>
        </ROW>
        <ROW>
                <NOME>Fortaleza</NOME>
        </ROW>
        <ROW>
                <NOME>Belo Horizonte</NOME>
        </ROW>
        <ROW>
                <NOME>Brasilia</NOME>
        </ROW>
        <ROW>
                <NOME>Curitiba</NOME>
        </ROW>
        <ROW>
                <NOME>Curitiba</NOME>
        </ROW>
        <ROW>
                <NOME>Manaus</NOME>
        </ROW>
        <ROW>
                <NOME>Recife</NOME>
        </ROW>
        <ROW>
                <NOME>Porto Alegre</NOME>
        </ROW>
</ROWSET>
```

Figura 44. Resultado da Procedure

6.2 IMPORTANDO DADOS XML DA TABELA CIDADES

Nesta sessão serão apresentados os processos de importação no Oracle 10g Express Edition e no Oracle 10g Standart.

6.2.1 Exportando Dados XML da Tabela Cidades No Oracle 10g Express Edition

O processo para importar dados XML é tão simples quando exportar, porém, é necessário que a tabela que receberá os dados importados, tenha a mesma estrutura do arquivo, ou seja, os mesmos campos.

Para importar o arquivo XML é necessário seguir os passos mostrados no processo de exportação até chegar na etapa de escolher entre Carga de Dados e Descarga de Dados, como ilustrado na Figura 41.

Figura 45. Carregar Dados no Banco de Dados

A opção Carregar disponibiliza outras três opções, das quais Carregar Dados XML será escolhida.

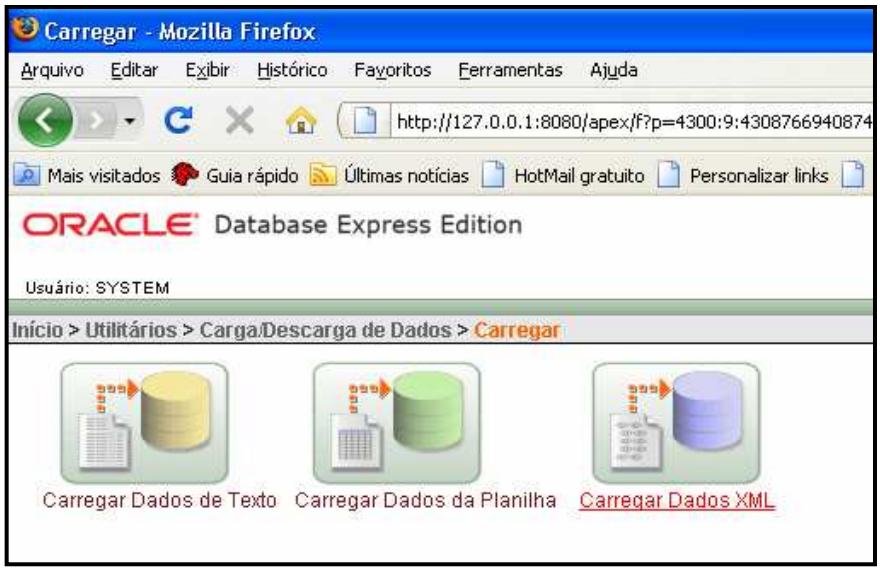

Figura 46. Carregar Dados XML no Banco de Dados

A próxima etapa é idêntica ao da exportação, é onde será feita a conexão com o esquema system onde se encontra a tabela “cidades_xml” para onde será importado o arquivo XML.

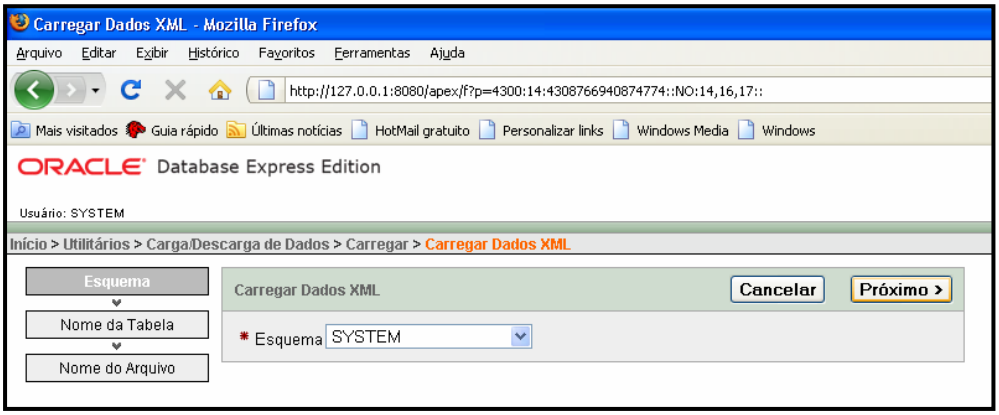

Figura 47. Conectar ao Esquema

A seguir abrirá uma janela onde podem ser visualizadas todas as tabelas do banco de dados. A tabela “cidades_xml” será selecionada.

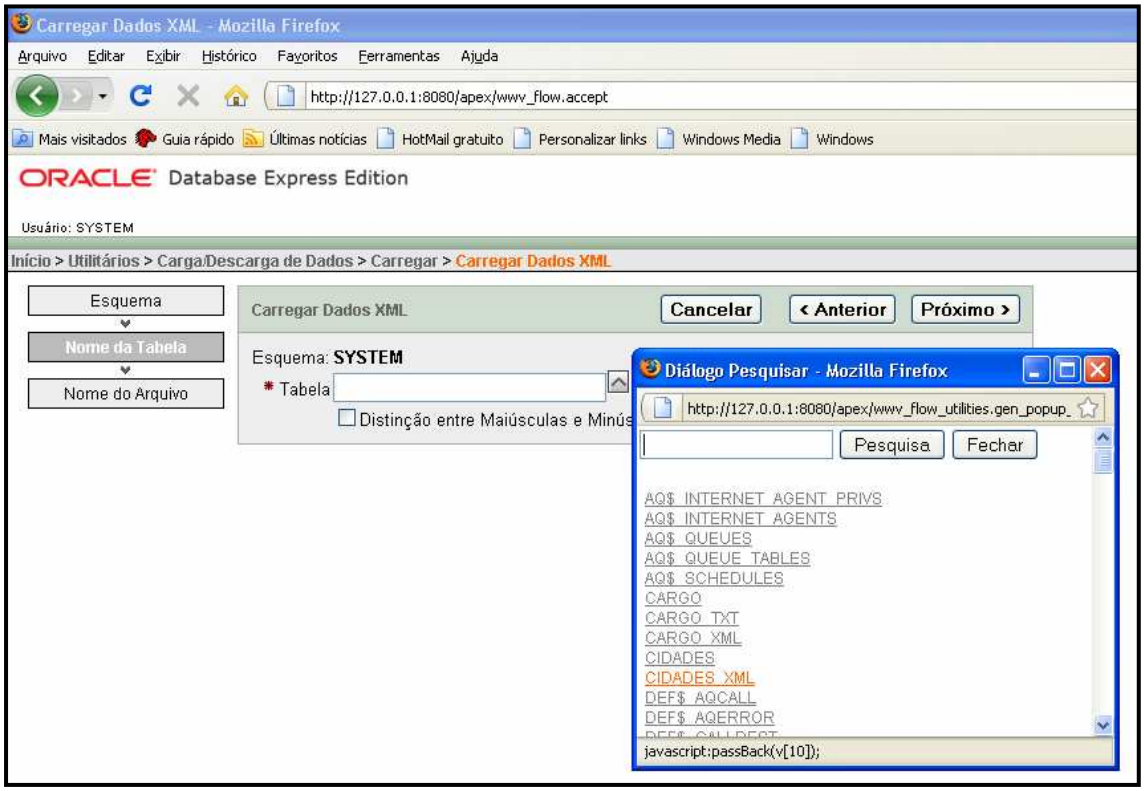

Figura 48. Selecionando a Tabela

A próxima etapa é responsável pela busca do arquivo XML a ser importado para a tabela cidades_xml.

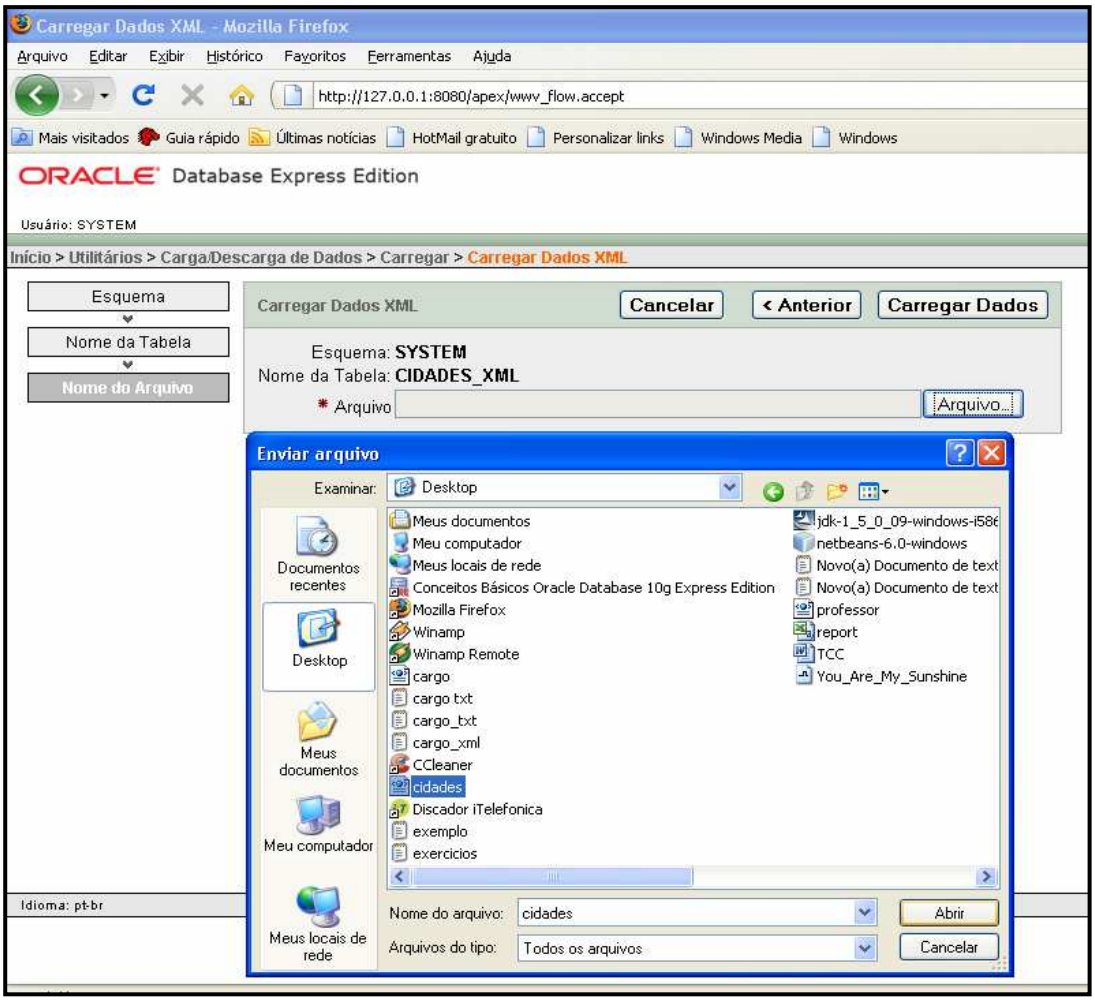

Figura 49. Selecionando a Arquivo XML

O arquivo XML é carregado e seus dados são importados para a tabela "cidades_xml".

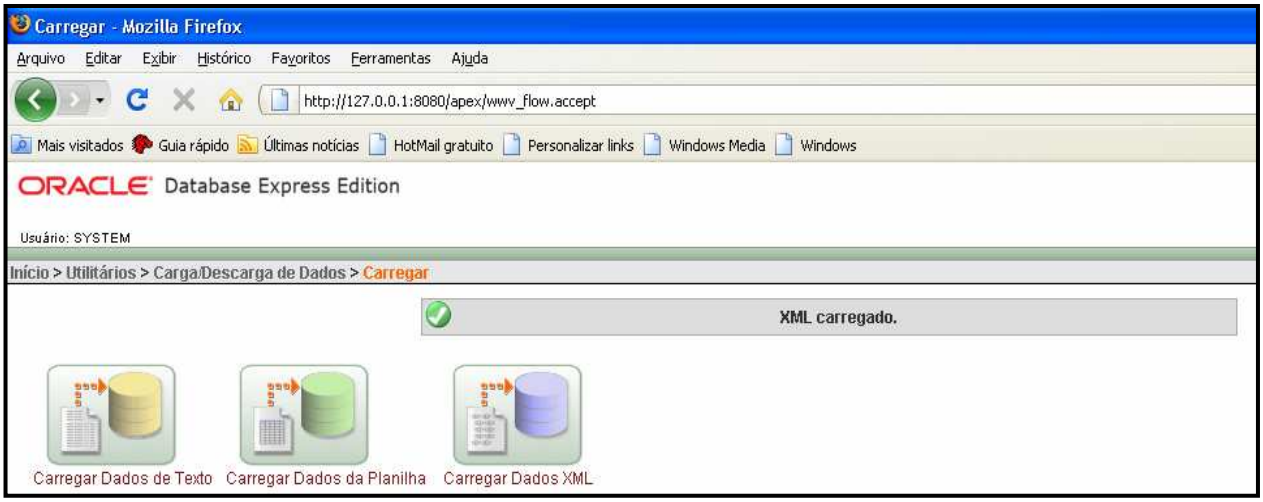

Figura 50. XML Carregado

6.2.2 Importando Dados XML da Tabela Cidades No Oracle 10g Standart

O processo de importação no Oracle 10g Standart, começa com a criação de uma tabela com a mesma estrutura do documento XML. No caso será criada uma tabela com o nome de cidades_xml.

A procedure utilizada para a importação dos dados XML para a tabela cidades_xml é ilustrada a seguir:

```
CREATE OR REPLACE PROCEDURE loadxml
(p_key number,
 p_tabela varchar2)
AS
fil    BFILE;
buffer RAW(32767);
len    INTEGER;
insrow INTEGER;
BEGIN
SELECT f_lob
INTO fil
FROM xml_temp
WHERE key = p_key;
DBMS_LOB.FILEOPEN(fil,DBMS_LOB.FILE_READONLY);
len := DBMS_LOB.GETLENGTH(fil);
DBMS_LOB.READ(fil,len,1,buffer);
xmlgen.resetOptions;
insrow :=
xmlgen.insertXML(p_tabela,UTL_RAW.CAST_TO_VARCHAR2(buffer));
DBMS_OUTPUT.PUT_LINE(insrow);
IF DBMS_LOB.FILEISOPEN(fil) = 1 THEN
DBMS_LOB.FILECLOSE(fil);
END IF;
EXCEPTION
WHEN OTHERS THEN
    DBMS_OUTPUT.PUT_LINE(sqlerrm);
    DBMS_OUTPUT.PUT_LINE(SQLERRM(SQLCODE));
IF DBMS_LOB.FILEISOPEN(fil) = 1 THEN
  DBMS_LOB.FILECLOSE(fil);
END IF;
end; /
```

Figura 51. Procedure para Importação XML

Para executar a procedure é preciso seguir o que mostra a figura a seguir:

```
set serveroutput on
exec loadxml(1,'mt_nota_fiscal_rc');
```

Figura 52. Executar a Procedure

Após esse passo, basta executar uma select na tabela cidades_xml para verificar os dados importados.

7 CONSIDERACÕES FINAIS E PROJEÇÕES FUTURAS

Este trabalho mostrou o porquê da utilização da Linguagem XML nos processos de exportação e importação de dados, ao invés dos tradicionais arquivos “.txt”, contornando assim, problemas encontrados com o uso de arquivos texto.

Em intercâmbio de dados realizado com documentos “.txt”, podem ocorrer problemas como falta de integridade dos dados pelo fato de que, esse tipo de documento pode ser facilmente violado. Outro problema encontrado nesse formato, é que o responsável por receber esses documentos “.txt”, fica totalmente preso ao layout, pois o documento não fornece uma estrutura onde pode-se compreender os elementos que o compõem, tornando complexo e trabalhoso o processo de importação. Já em intercâmbios de dados com arquivos em formato XML, esse problema é facilmente contornado, pois o mesmo é auto-documentável e com isso, sua estrutura define claramente cada tipo de elemento nele contido.

XML é frequentemente utilizado na web, pois disponibiliza dados de forma estruturada e padronizada, facilitando assim a troca de informações entre organizações.

A proposta de trabalho aqui apresentada, foi voltada especificamente para a exportação e importação de dados entre tabelas, onde no estudo de caso cada fase desses processos foram ilustrados e explicados. A XML possui várias funções que não foram exploradas e focalizadas neste trabalho, proporcionando assim que as mesmas possam ser estudadas em trabalhos futuros.

Dessa forma, este trabalho veio a contribuir com a ampla área de pesquisa em bancos de dados, servindo como material de referência para possíveis trabalhos futuros, onde poderão ser exploradas e aplicadas outras ferramentas da linguagem XML.

8 REFERÊNCIAS BIBLIOGRÁFICAS

ALMEIDA, Maurício Barcellos. Uma introdução ao XML, sua utilização na Internet e alguns conceitos complementares. **Ci. Inf.**, Brasília, v.31, n.2, p. 5-13, maio/agosto 2002.

BARCAROLI, Velcir. **Análise Comparativa de Linguagens de Consulta para Banco de Dados XML Nativo.** 2005. 46 p. – Universidade Federal de Santa Catarina, Florianópolis, 2005.

CHEN, Whei-Jen; SAMMARTINO, Art; GOUTEV, Dobromir; HENDRICKS, Felicity; KOMI, Ippei; WEI, Ming-Pang; AHUJA, Rav. **DB2 9 pureXML Guide.** IBM. – Janeiro : Redbooks, 2007.

GRANDI, Fabio; TIBERIO, Paolo; MANDREOLI, Federica, BERGINZINI, Marco. **A Temporal Data Model and Management System for Normative Texts in XML Format. –** New Orleans, Louisiana, USA, 2003.

GRAVES, Mark. **Projeto de Banco de Dados com XML.** Tradução de Aldair José Coelho da Silva. São Paulo: Person Education do Brasil, 2003.

GARCIA, José Roberto M; NING, Dr Carlos Ho Shih. Relacionando documentos XML a um esquema de banco de dados relacional. **Anais do WORCAP, INPE**, São José dos Campos, 25 de Outubro de 2001, p.30-32.

JUNIOR, Miguel Benedito Furtado. **XML – Extensible Markup Language.** Universidade Federal do Rio de Janeiro – UFRJ. Disponível em <http://www.gta.ufrj.br/grad/00_1/miguel/link8.htm>. Acesso em: 14 mai. 2008.

MARTINS, Juliano Marcos, POLETTO, Alex Sandro Romeu de Souza. DB2 pureXML: Entendendo e Aplicando. **IBM Brasil**, Hortolândia, São Paulo; Fundação Educacional do Município de Assis, Assis, São Paulo.

MASSARO,Giuliano Alves; FORNARI, Miguel Rodrigues. **Evolução de Esquemas e Documentos XML no Oracle XML DB.** Universidade Luterana do Brasil - ULBRA, Canoas, RS, Brasil.

MARQUES, Eduardo Feltrin; ZAUPA, Aglaê Pereira. **O uso do XML na Integração de Banco de Dados Relacionais.** Universidade do Oeste Paulista – UNOESTE, Faculdade de Informática de Presidente Prudente – FIPP. Disponível em <http://www.webartigos.com/articles/2762/1/o-uso-do-xml-na-integracao-de-banco-de-dados-relacionais/pagina1.html>. Acesso em: 02 jun. 2008.

PINTO, Marcus Barbosa; SACCOL, Deise de Brum. Um estudo sobre esquemas para documentos XML. **Anais do V Encontro de Estudantes de Informática do Tocantins.** Palmas, TO, Outubro, 2003, p. 211-220.

PRICE, Jason. **Oracle Database 11g SQL.** Tradução de João Eduardo Nóbrega Tortello. Porto Alegre : Bookmark, 2009.

RICARTE, Ivan Luiz Marques. **Banco de Dados Relacionais.** UNICAMP. Disponível em <http://www.dca.fee.unicamp.br/cursos/PooJava/javadb/bdrel.html>. Acesso em: 20 mai.2008.

SILBERSCHATZ, Abraham; KORTH, Henry F. **Sistema de Banco de Dados.** Tradução de Maurício Heihachiro Galvan Abe. 2ª edição – São Paulo: Makron, 1993.

SILBERSCHATZ, Abraham; KORTH, Henry F.; SUDARSHAN, S. **Sistemas de Banco de Dados.** Tradução de Daniel Vieira. 5ª edição – Rio de Janeiro : Elsevier, 2006.

SILVA, Rosane Gagliardi. **Uma Proposta de Extensão Temporal para XML e a Realização de Consultas.** – Universidade Federal do Rio Grande do Sul - Porto Alegre.

SANTOS, Rodrigo Gasparoni; EDELWEISS, Nina; GALANTE, Renata de Matos. **Evolução de Documentos XML com Tempo e Versões. –** Universidade Federal do Rio Grande do Sul – Porto Alegre.

TITTEL, Ed**. Teoria e Problemas de XML.** Tradução de Ralph Miller Jr.; Cons. Sup. e Rev. Téc. Marcelo Soares Pimenta. – Porto Alegre : Bookman, 2003.

VIEIRA,Humberto; RUBERG, Gabriela; MATTOSO, Marta. XVertet: Armazenamento e Consulta de Dados XML em SGBDs. **XVII Simpósio Brasileiro de Banco de Dados.** Universidade Federal do Rio de Janeiro - UFRJ, Rio de Janeiro, Brasil.

WANG, Fusheng; ZHOU, Xin; ZANIOLO, Carlo. **Bridging Relational Database History and the Web: the XML Approach. –** Arlington, Virginia, USA, November, 2006.