Gerenciamento de Dados e Informação

Fernando Fonseca
Ana Carolina
Robson Fidalgo

Centro de Informática — Universidade Federal de Pernambuco — CIn.ufpe.br

Suporte Nativo a XML no Oracle 10g

ORACLE DATABASE 10g Oracle XML DB

Oracle XML DB

- Conjunto de tecnologias do SGBD Oracle relacionadas ao alto desempenho em armazenamento e recuperação de dados XML
  - Provê suporte nativo a XML tratando os modelos de dados de SQL e XML de uma maneira interoperável
- Características
  - Suporte aos modelos de dados de XML e XML Schema (incorporados ao SGBD) como definido pelo W3C e a métodos de acesso padrão para navegar e consultar dados XML

Oracle XML DB

- Características (Cont.)
  - Modos de armazenar, consultar, atualizar e transformar (uso de XSLT) dados XML acessando-os por meio de SQL
  - Modos de realizar operações XML em dados SQL
  - Um repositório simples e leve para XML no qual se pode organizar e gerenciar conteúdo de BD, incluindo XML, usando uma metáfora arquivo/diretório/URL

Oracle XML DB

- Características (Cont.)
  - Uma estrutura para armazenar e gerenciar dados XML, independente de forma de armazenamento, conteúdo e linguagem de programação
    - Gerenciamento de hierarquias de documentos XML
  - Modos padrão da indústria para acessar e atualizar dados XML
    - XPath da W3C
    - Padrão ISO-ANSI SQL/XML

Oracle XML DB

- Características
  - Padrão da indústria (Cont.)
    - Entrada e saída de conteúdo XML para/do BD podem ser realizadas por FTP, HTTP(S), e WebDAV
    - API padrão provê acesso por programas e manipulação de conteúdo XML com Java, C e PL/SQL
  - Gerenciamento de memória e otimizações específicas para XML
  - Confiabilidade, disponibilidade, escalabilidade e segurança

Oracle XML DB

- Tipo pré-definido para criar, extrair, indexar e validar dados XML com XML Schema: **XMLType**
  - Coluna ou Tabela
  - Armazena o conteúdo do documento como texto XML utilizando o tipo de dado Character Large Object (CLOB) – armazenamento **não-estruturado** - ou como conjuntos de **objetos** – armazenamento **estruturado**
- Principais benefícios
  - União dos mundos: XML e SQL
  - Indexação e navegação
  - *Parser* embutido
  - Combinação de XMLType com outros tipos

7

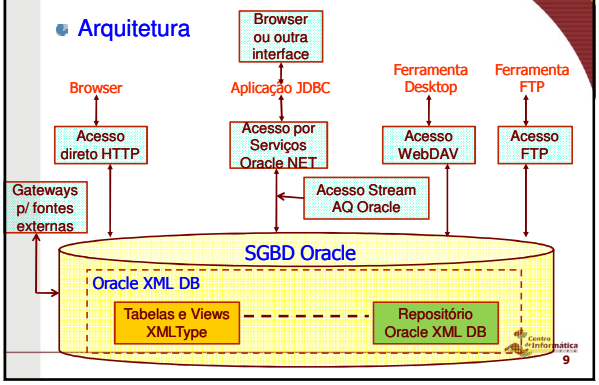
Oracle XML DB

- Quando usar XMLType
  - Armazenar e recuperar documento XML como um todo
  - Consultas em elementos XML
  - Utilizar a funcionalidade XPath
  - Preparar para futuras atualizações

8

Oracle XML DB

- Arquitetura



9

Oracle XML DB

- Stream AQ Oracle
  - Provê o gerenciamento de mensagens e a comunicação necessária para integração de aplicações
- WebDAV (Web-based Distributed Authoring and Versioning)
  - Um conjunto de **extensões** ao protocolo do HTTP que permite aos usuários **editar** e **gerenciar** arquivos **colaborativamente** em servidores Web **remotos**

10

Oracle XML DB

- Serviços XML em tabelas e views XMLType
  - Validação
  - Transformação (Uso de XSLT)
  - Registro de XML Schema
  - Create Table (armazenamento LOB ou O-R)
  - Insert, Delete, Update em tabelas XMLType
  - Indexação

11

Oracle XML DB

- Recuperar / Gerar XML usando API XML Type
  - SQL
  - Java
  - PL/SQL
  - C
  - C++

12

Oracle XML DB

- Serviços XML em repositórios XML DB
  - Versão
  - Segurança ACL (Access Control List)
  - Diretórios
- Recuperar / Gerar XML usando recursos das API
  - SQL
  - Java
  - PL/SQL

13

Oracle XML DB

- Armazenamento de XMLType
  - Quando XML Schemas são registrados no Oracle XML DB um conjunto de tabelas default é criado e usado para armazenar instâncias de documentos XML associados a cada schema
  - Esses documentos também podem ser vistos e acessados no repositório Oracle XML DB

14

Oracle XML DB

- Tabelas e colunas XMLType podem ser restringidas de acordo com um XML Schema
  - O SGBD garantirá que somente os documentos XML que sejam validados de acordo com o XML Schema possam ser armazenados na coluna ou na tabela
  - Desde que o conteúdo da tabela esteja de acordo com uma estrutura conhecida de XML, o Oracle XML DB pode usar a informação contida no XML schema para fornecer processamento de consultas ou de atualizações mais inteligente de dados XML

15

Oracle XML DB

- Tabelas e colunas XMLType podem ser restringidas de acordo com um XML Schema (Cont.)
  - Restrigir o XMLType a um XML Schema fornece a opção de armazenar o conteúdo do documento usando técnicas de armazenamento estruturado. Este decompõe o conteúdo do documento XML e o armazena como um conjunto de objetos SQL em vez de simplesmente armazenar o documento como texto em um CLOB. O modelo de objeto usado para armazenar o documento é derivado automaticamente do conteúdo do XML schema

16

Oracle XML DB

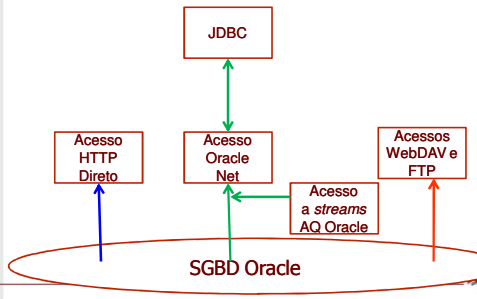
- Validação de conteúdo de um XMLType de acordo com um XML Schema
  - Uso do método

schemavalidate( )

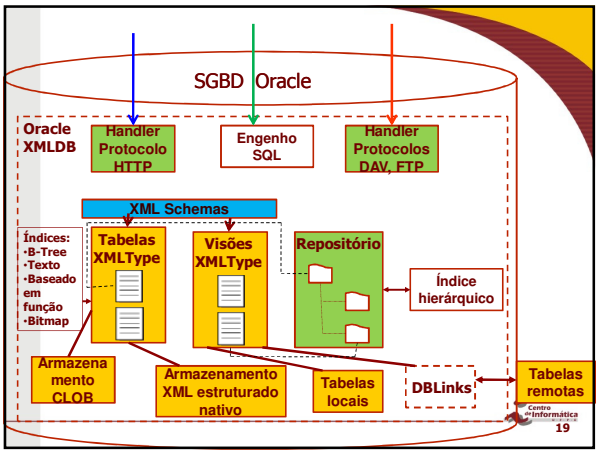
17

Oracle XML DB

- Armazenamento de XMLType



18



Recursos

- XML Parsers
  - Suporte a interfaces DOM e SAX
- XML Class Generator
  - Código fonte a partir de XML DTD ou Schema
- XML SQL Utility
  - Gerar documento XML a partir de consulta SQL
    - DBMS_XMLGEN, SYS_XMLGEN, ...

20

Usando XML

- Criando tabela com atributo XML

Coluna SYS.XMLTYPE

```
CREATE TABLE lojas(
loja_id NUMBER(5),
loja_desc SYS.XMLTYPE,
loja_nome VARCHAR2(35),
localizacao_id NUMBER(4)
);
```

21

Usando XML

- Inserindo dados XML
  - Função `createXML()` para instanciação

Função createXML( )

```
INSERT INTO lojas (loja_id, loja_desc)
 VALUES (1001, sys.XMLType.createXML(
      ' <Loja LjNo=" 100">
          <Predio>Próprio</Predio>
        </Loja>' ));
```

22

Usando XML

- Consultando XML

Expressão de caminho - XPath

```
SELECT
    l.loja_desc.extract('/Loja/Predio/text( ) ').getStringVal( )
      "Predio"
FROM lojas l;
```

Função extract( )

23

Usando XML

- Atualizando XML

Função createXML( )

```
UPDATE lojas SET loja_desc =
        Sys.XMLType.createXML(
              '<Loja ljNo= " 200 " >
                  <Predio>Alugado</Predio>
               </Loja>') WHERE loja_id = 1004;
```

24

Usando XML

- Removendo XML

```
DELETE FROM lojas l
  WHERE l.loja_desc.extract('//Predio/text( ) ').getStringVal( )
            = ' Alugado';
```

Função extract( )

Exemplo

- Considerando a tabela Lojas
- Inserindo os dados abaixo

```
INSERT INTO lojas (loja_id, loja_desc)
  VALUES (1001, sys.XMLType.createXML( ' <Loja LjNo="100">
            <Predio>Próprio</Predio>  </Loja>' ));

INSERT INTO lojas (loja_id, loja_desc)
  VALUES (1002, sys.XMLType.createXML( ' <Loja LjNo="200">
            <Predio>Alugado</Predio>  </Loja>' ));

INSERT INTO lojas (loja_id, loja_desc)
  VALUES (1003, sys.XMLType.createXML( ' <Loja LjNo="300">
            <Predio>Comodato</Predio> </Loja>' ));

INSERT INTO lojas (loja_id, loja_desc)
  VALUES (1004, sys.XMLType.createXML(<Loja LjNo="400">
            <Predio>Leasing</Predio> </Loja>' ));
```


Exemplo

- Realizando a consulta

```
SELECT
    l.loja_desc.extract('/Loja/Predio/text( )').getStringVal( )
      "Predio"
FROM lojas l;
```

- Resultado

Predio
Próprio
Alugado
Comodato
Leasing

Exemplo

- Realizando a consulta

```
SELECT
    l.loja_desc.extract('/Loja/Predio/text( )').getStringVal( )
      "Predio", l.loja_id
FROM lojas l;
```

- Resultado

Predio	LOJA_ID
Próprio	1001
Alugado	1002
Comodato	1003
Leasing	1004

Exemplo

- Executando o comando

```
UPDATE lojas SET loja_desc =
          Sys.XMLType.createXML(
                '<Loja ljNo="300">
                     <Predio>Alugado</Predio>
                </Loja>')  where loja_id = 1004;
```


Exemplo

- Repetindo a consulta

```
SELECT
    l.loja_desc.extract('/Loja/Predio/text( )').getStringVal( )
      "Predio" , l.loja_id
FROM lojas l;
```

- Resultado

Predio	LOJA_ID
Próprio	1001
Alugado	1002
Comodato	1003
Alugado	1004

Exemplo

- Executando o comando

```
DELETE FROM lojas l
  WHERE l.loja_desc.extract('//Predio/text( )').getStringVal( )
              = 'Alugado';
```


Exemplo

- Repetindo a consulta

```
SELECT
    l.loja_desc.extract('/Loja/Predio/text( )').getStringVal( )
      "Predio"
FROM lojas l;
```

- Resultado

Predio
Próprio
Comodato

Outras funcionalidades

- Considerar a tabela

```
CREATE TABLE carro(
    chassi VARCHAR2(20) NOT NULL,
    modelo VARCHAR2(30) NOT NULL,
    data_carro DATE NOT NULL,
    km_carro INTEGER NOT NULL,
    CONSTRAINT carro_pk
        PRIMARY KEY(chassi)
);
```


Outras funcionalidades

- Inserindo dados

```
INSERT INTO carro(chassi, modelo, data_carro, km_carro)
VALUES ('1', 'Clio', to_date('22/03/2007','DD/MM/YYYY'),
1231);
INSERT INTO carro(chassi, modelo, data_carro, km_carro)
VALUES ('2', 'Audi A4', to_date('30/04/2007','DD/MM/YYYY'),
1245);
INSERT INTO carro(chassi, modelo, data_carro, km_carro)
VALUES ('3', 'Scenic', to_date('23/02/2007','DD/MM/YYYY'),
1000);
INSERT INTO carro(chassi, modelo, data_carro, km_carro)
VALUES ('4', 'Gol', to_date('01/03/2007','DD/MM/YYYY'), 700);
```


Outras funcionalidades

- Função SYS_XMLGEN ( )
  - Usada para gerar XML dentro de consultas SQL
  - Mistura elementos de XML com SQL
  - Retorna um tipo XMLType
- Realizando a consulta

```
SELECT SYS_XMLGEN(modelo) as XML FROM carro;
```


Outras funcionalidades

- Resultado

XML
<?xml version="1.0"?> <MODELO>Clio</MODELO>
<?xml version="1.0"?> <MODELO>Audi A4</MODELO>
<?xml version="1.0"?> <MODELO>Scenic</MODELO>
<?xml version="1.0"?> <MODELO>Gol</MODELO>

Outras funcionalidades

- Pacote DBMS_XMLGEN
  - Cria documentos XML a partir de consultas SQL
- Um roteiro de uso
  - Permitir output na interface de caracteres
    - Set serveroutput on;
  - Definir um bloco
    - Declarar variável para criar um contexto
      - ctx dbms_xmlgen.ctxhandle;

Outras funcionalidades

- Definir um bloco (Cont.)
  - Declarar variável do tipo CLOB para armazenar o arquivo XML gerado
    - result clob;
  - Criar um novo contexto com a consulta SQL apropriada
    - ctx := dbms_xmlgen.newContext('select * from carro');

Outras funcionalidades

- Definir um bloco (Cont.)
  - Personalizar a tag de entidade
    - DBMS_XMLGEN.setRowTag(ctx,'CARRO');
  - Gerar um valor CLOB como resultado
    - result := dbms_xmlgen.getXML(ctx);
  - Dar saída do resultado
    - dbms_output.put_line(result);
  - Fechar o contexto
    - dbms_xmlgen.closeContext(ctx);

Outras funcionalidades

- Exemplo de bloco

```
set serveroutput on;
declare
  ctx dbms_xmlgen.ctxhandle;
  result clob;
begin
   -- criar um novo contexto para a consulta SQL
   ctx := dbms_xmlgen.newContext('select * from carro');
  -- personalizar a tag de entidade
 DBMS_XMLGEN.setRowTag(ctx,'CARRO');
   -- gerar um valor  CLOB como resultado
   result := dbms_xmlgen.getXML(ctx);
  -- dar saída do resultado
  dbms_output.put_line(result);
  -- fechar o contexto
  dbms_xmlgen.closeContext(ctx);
end;
/
```


Outras funcionalidades

- Resultado

```
<?xml version="1.0"?>
<ROWSET>
 <CARRO>
  <CHASSI>1</CHASSI>

<MODELO>Clio</MODELO>
  <DATA_CARRO>22/03/07</DATA_CARRO>

<KM_CARRO>1231</KM_CARRO>
 </CARRO>
 <CARRO>
  <CHASSI>2</CHASSI>
  <MODELO>Audi A4</MODELO>
  <DATA_CARRO>30/04/07</DATA_CARRO>
  <KM_CARRO>1245</KM_CARRO>

</CARRO>
```


Outras funcionalidades

- Resultado (Cont.)

```
<CARRO>
  <CHASSI>3</CHASSI>
  <MODELO>Scenic</MODELO>

<DATA_CARRO>23/02/07</DATA_CARRO>
  <KM_CARRO>1000</KM_CARRO>
 </CARRO>

<CARRO>
  <CHASSI>4</CHASSI>
  <MODELO>Gol</MODELO>

<DATA_CARRO>01/03/07</DATA_CARRO>
  <KM_CARRO>700</KM_CARRO>
 </CARRO>
```

Outras funcionalidades

- Resultado (Cont.)

```
<CARRO>

<CHASSI>5</CHASSI>
  <MODELO>Ferrari</MODELO>

<DATA_CARRO>11/01/07</DATA_CARRO>
  <KM_CARRO>1290</KM_CARRO>

</CARRO>
</ROWSET>
```


Exercício Extra

- Gerar arquivo XML a partir de consulta em uma das tabelas do modelo OR implementado pela equipe no Oracle
- Criar uma DTD apropriada para o arquivo XML
  - Não usar a cláusula ANY
- Criar uma formatação CSS apropriada para o arquivo gerado – Mostrar dados de tabelas na forma de tabela
- Exibir o arquivo no browser
- Data de entrega: junto com a apresentação do projeto objeto-relacional de cada equipe

Até **1,0** ponto para passar por média!