Gerenciamento de Dados e Informação

Suporte Nativo a XML no Oracle

Fernando Fonseca
Ana Carolina
Robson Fidalgo

Oracle & XML

ORACLE DATABASE 11g

- Habilita uma fonte confiável para XML
- Apresenta flexibilidade para permitir ótimo processamento de XML tanto centrado em dados (data-centric) quanto em conteúdo (content-centric)
- Mantém o compromisso da Oracle com a confiabilidade, segurança, disponibilidade e escalabilidade
- Implementa as características chave do padrão XML

Oracle & XML

- Dá suporte aos paradigmas de desenvolvimento centrado em SQL (SQL-centric), em XML (XML-centric) e em documentos (document-centric)
- Dá suporte a XML nos servidores de banco de dados e de aplicações
- Inovação sustentável

Binary XML Storage & Indexing

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 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 independente de armazenamento, conteúdo e linguagem de programação para armazenar e gerenciar dados XML
    - 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

Oracle XML DB

- Tipo pré-definido para criar, extrair, indexar e validar dados XML com XML Schema: **XMLType**
  - Pode ser aplicado a 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

Oracle XML DB

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

Oracle XML DB

- Tabelas e colunas XMLType podem ser restringidas de acordo com um XML Schema (Cont.)
  - Restringir o XMLType a um XML Schema
    - Fornece a opção de armazenar o conteúdo do documento usando técnicas de armazenamento estruturado
      - 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

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)
);
```

Armazenamento CLOB (Não estruturado)

Usando XML

- Criando tabela XML como objeto (Binário)
  - Duas formas
    - Como uma tabela XMLTYPE

```
CREATE TABLE clientes OF XMLTYPE;
```

Usando XML

- Criando tabela XML como objeto (Binário) (Cont.)
  - Solução objeto-relacional

```
CREATE TABLE lojas (
loja_id NUMBER(5),
loja_desc XMLTYPE,
loja_nome VARCHAR2(35),
localizacao_id NUMBER(4)
)
XMLTYPE column loja_desc
XMLSCHEMA "http://www.cin.ufpe.br/~fdfd/CadeiaLojas.xsd"
ELEMENT "CadeiaLojas"
STORE AS OBJECT RELATIONAL;
```


Usando XML

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

schemavalidate( )

Usando XML

- Registro de esquema XML com armazenamento objeto-relacional

```
DBMS_XMLSCHEMA.registerSchema (
SCHEMAURL => 'http://www.cin.ufpe.br/~fdfd/
                    CadeiaLojas.xsd',
SCHEMADOC => xmlType(bfilename('XMLDIR','cl.xsd'),
nls_charset_id('AL32UTF8')),
GENTYPES => TRUE,
GENTABLES => TRUE )
```


Usando XML – Armazenamento CLOB

- 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>' ));
```


Usando XML – Armazenamento CLOB

- Consultando XML

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

Função extract( )

Usando XML – Armazenamento CLOB

- Atualizando XML

Função createXML( )

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

Usando XML – Armazenamento CLOB

- Removendo XML

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

Função extract( )

Exemplo

- Considerando a tabela Lojas como definido abaixo

```
CREATE TABLE lojas(
loja_id NUMBER(5),
loja_desc SYS.XMLTYPE
);
```

- Inserindo os dados

```
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 consulta SQL convencional

```
SELECT * FROM lojas;
```

- Resultado

LOJA_ID	LOJA_DESC
1001	<Loja LjNo="100"> <Predio>Próprio</Predio> </Loja>
1002	<Loja LjNo="200"> <Predio>Alugado</Predio> </Loja>
1003	<Loja LjNo="300"> <Predio>Comodato</Predio> </Loja>
1004	<Loja LjNo="400"> <Predio>Leasing</Predio> </Loja>

Exemplo

- Realizando a consulta – manipulação XML

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

Expressão XPATH

- 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

Removidas as duas tuplas com informação sobre Prédio = Alugado

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);
INSERT INTO carro(chassi, modelo, data_carro, km_carro)
VALUES ('5', 'Ferrari', to_date('11/01/2007','DD/MM/YYYY'), 1290);
```


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;
```

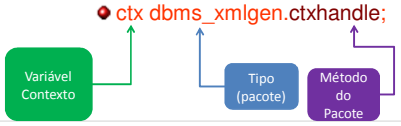
Outras funcionalidades

- 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>
<?xml version="1.0"?> <MODELO>Ferrari</MODELO>

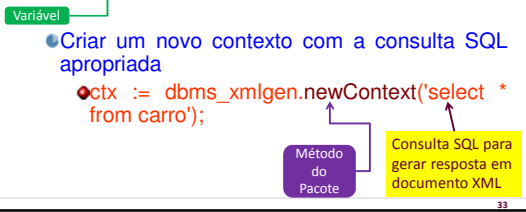
Outras funcionalidades

- 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');

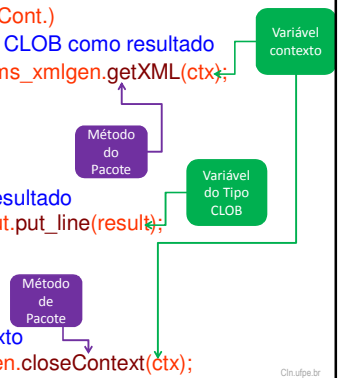


Outras funcionalidades

- Definir um bloco (Cont.)
  - Personalizar a tag raiz do documento
    - DBMS_XMLGEN.setRowset(ctx, 'Locadora');
  - Personalizar a tag de entidade
    - DBMS_XMLGEN.setRowTag(ctx,'CARRO');

Outras funcionalidades

- Definir um bloco (Cont.)
  - 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 as tags raiz e de entidade
DBMS_XMLGEN.setRowsetTag(ctx, 'LOCADORA');
 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"?>
<LOCADORA>
 <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>
```

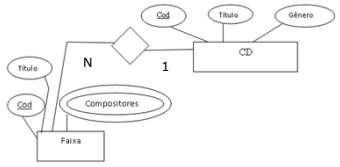

Outras funcionalidades

- Resultado (Cont.)

```
<CARRO>

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

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

</CARRO>
</LOCADORA>
```


Outras funcionalidades

- Considerando o contexto objeto-relacional
  - Dado o modelo ERE



Outras funcionalidades

- Criação de Tipos


```
CREATE OR REPLACE TYPE tp_compositor AS OBJECT (
nome VARCHAR2(16) );
/
```

```
CREATE OR REPLACE TYPE tp_compositores AS VARRAY(3) OF
tp_compositor;
/
```


Outras funcionalidades

- Criação de Tipos (Cont.)


```
CREATE OR REPLACE TYPE tp_faixa AS OBJECT(
  cod integer,
  titulo  varchar2(23),
  compositores tp_compositores
) ;
/
```

Outras funcionalidades

- Criação de Tipos (Cont.)

```
CREATE OR REPLACE TYPE tp_ref_relac AS OBJECT(
  faixa REF tp_faixa);
/
```

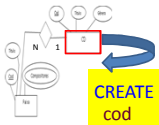
```
CREATE TYPE tp_nt_ref_relac  AS TABLE OF tp_ref_relac;
/
```

CIn.ufpe.br

43

Outras funcionalidades

- Criação de Tipos (Cont.)



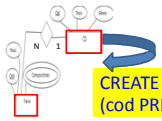
```
CREATE OR REPLACE TYPE tp_cd AS OBJECT (
 cod            INTEGER,
 titulo         VARCHAR2(20),
 genero         VARCHAR2(12),
 faixas          tp_nt_ref_relac
);
/
```

CIn.ufpe.br

44

Outras funcionalidades

- Criação de Tabelas



```
CREATE TABLE tb_faixa  OF tp_faixa
(cod PRIMARY KEY);
```

```
CREATE TABLE tb_cd  OF tp_cd
(cod PRIMARY KEY)
NESTED TABLE faixas STORE AS lista_faixas;
```

CIn.ufpe.br

45

Outras funcionalidades

- Inserir dados – tb_faixa

```
INSERT INTO tb_faixa VALUES
        (tp_faixa(1, 'Nossa Canção', tp_compositores(
                        tp_compositor('Roberto Carlos'),
                        tp_compositor('Erasmo Carlos') )));
```

```
INSERT INTO tb_faixa VALUES
        (tp_faixa(2, 'Negro Gato', tp_compositores(
                        tp_compositor('Roberto Carlos'),
                        tp_compositor('Erasmo Carlos'),
                        tp_compositor('Rossini Pinto')  )));
```

CIn.ufpe.br

46

Outras funcionalidades

- Inserir dados – tb_faixa (Cont.)

```
INSERT INTO tb_faixa VALUES
        (tp_faixa(3, 'Copacabana', tp_compositores(
                        tp_compositor('Antonio Maria') )));
```

```
INSERT INTO tb_faixa VALUES
        (tp_faixa(4, 'Desafinado', tp_compositores(
                        tp_compositor('João Gilberto') )));
```

```
INSERT INTO tb_faixa VALUES
        (tp_faixa(5, 'Todo mundo gosta de mim', tp_compositores(
                        tp_compositor('Ronaldo Boscoli'),
                        tp_compositor('Nelson Mota'),
                        tp_compositor('Tom Zé'))));
```

CIn.ufpe.br

47

Outras funcionalidades

- Inserir dados – tb_cd

```
INSERT INTO tb_cd VALUES
        (1, 'Perfil', 'Coletânea', tp_nt_ref_relac (
                tp_ref_relac((SELECT REF(F) FROM tb_faixa F  WHERE
                    F.cod =1)),
                tp_ref_relac((SELECT REF(F) FROM tb_faixa F  WHERE
                    F.cod =2))));
```

```
INSERT INTO tb_cd VALUES
        (2, 'Páginas da Vida', 'Novela', tp_nt_ref_relac (
                tp_ref_relac((SELECT REF(F) FROM tb_faixa F  WHERE
                    F.cod =3)),
                tp_ref_relac((SELECT REF(F) FROM tb_faixa F  WHERE
                    F.cod =4))));
```

48

Outras funcionalidades

- Inserir dados – tb_cd (Cont.)

```
INSERT INTO tb_cd VALUES
        (3, 'Belíssima', 'Novela', tp_nt_ref_relac (
            tp_ref_relac((SELECT REF(F) FROM tb_faixa F  WHERE
                F.cod =5)) ));
```

- Consulta – tb_cd

```
SELECT * FROM  tb_cd;
```


Outras funcionalidades

- Resposta da Consulta

COD	TITULO	GENERO	FAIXAS(FAIXA)
1	Perfil	Coletânea	TP_NT_REF_RELAC(TP_REF_RELAC(0000220208241F5C619D99F743E05015AC07026337241F5C619D96F743E05015AC07026337), TP_REF_RELAC(000022020824 1F5C619D9AF743E05015AC07026337241F5C619D96F743E05015AC07026337))
1	Páginas da Vida	Novela	TP_NT_REF_RELAC(TP_REF_RELAC(0000220208241F5C619D9BF743E05015AC07026337241F5C619D96F743E05015AC07026337), TP_REF_RELAC(000022020824 1F5C619D9CF743E05015AC07026337241F5C619D96F743E05015AC07026337))
3	Belíssima	Novela	TP_NT_REF_RELAC(TP_REF_RELAC(0000220208241F5C619D9DF743E05015AC07026337241F5C619D96F743E05015AC07026337))

Outras funcionalidades

- Gerar documento XML contendo os títulos dos CD – Uso de SYS_XMLGEN

```
SELECT SYS_XMLGEN(titulo) as XML FROM tb_cd;
```

- Resposta da Consulta

```
XML
--------------------------------
<?xml version="1.0"?>
<TITULO>Perfil</TITULO>

<?xml version="1.0"?>
<TITULO>Páginas da Vida</TITULO>

<?xml version="1.0"?>
<TITULO>Belíssima</TITULO>
```


Outras funcionalidades

- Gerar documento XML contendo o resultado da consulta select * from tb_cd – Uso de DBMS_XMLGEN

```
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 tb_cd');
-- 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>
 <ROW>
  <COD>1</COD>
  <TITULO>Perfil</TITULO>

<GENERO>Coletânea</GENERO>
  <FAIXAS>
   <TP_REF_RELAC>

<FAIXA>0000220208241F5C619D99F7
43E05015AC07026337241F5C619D96
F743E05015AC07026337</FAIXA>

</TP_REF_RELAC>
```

Cod, Titulo, Genero, Faixas

Outras funcionalidades

- Resultado (Cont.)

```
<TP_REF_RELAC>

<FAIXA>0000220208241F5C619D9AF7
43E05015AC07026337241F5C619D96
F743E05015AC07026337</FAIXA>

</TP_REF_RELAC>
 </FAIXAS>
 </ROW>
 <ROW>
  <COD>2</COD>
  <TITULO>Páginas da Vida</TITULO>

<GENERO>Novela</GENERO>
```

Outras funcionalidades

- Resultado (Cont.)

```
<FAIXAS>
  <TP_REF_RELAC>

<FAIXA>0000220208241F5C619D9BF7
43E05015AC07026337241F5C619D96
F743E05015AC07026337</FAIXA>

</TP_REF_RELAC>
  <TP_REF_RELAC>

<FAIXA>0000220208241F5C619D9CF7
43E05015AC07026337241F5C619D96
F743E05015AC07026337</FAIXA>
```


Outras funcionalidades

- Resultado (Cont.)

```
</TP_REF_RELAC>
 </FAIXAS>
</ROW>

<ROW>
 <COD>3</COD>
 <TITULO>Belíssima</TITULO>

<GENERO>Novela</GENERO>
 <FAIXAS>
  <TP_REF_RELAC>

<FAIXA>0000220208241F5C619D9DF7
43E05015AC07026337241F5C619D96
F743E05015AC07026337</FAIXA>
```


Outras funcionalidades

- Resultado (Cont.)

```
</TP_REF_RELAC>
 </FAIXAS>
</ROW>
</ROWSET>
```


Outras funcionalidades

- Gerar documento XML contendo o resultado da consulta para retornar titulo do CD e o título e compositores de cada faixa – Uso de DBMS_XMLGEN

```
set serveroutput on;
declare
 ctx dbms_xmlgen.ctxhandle;
 result clob;
begin
  -- criar um novo contexto para a consulta SQL
  ctx := dbms_xmlgen.newContext('select c.titulo, f.faixa.titulo,
          f.faixa.compositores from tb_cd c, TABLE(c.faixas)f');
 -- personalizar as tags raiz e de entidade
DBMS_XMLGEN.setRowsetTag(ctx, 'Discoteca');
 DBMS_XMLGEN.setRowTag(ctx,'CD');
```


Outras funcionalidades

- Gerar documento XML (Cont.)

```
-- 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

Titulo, faixa.Titulo, faixa.compositores

```
<?xml version="1.0"?>
<Discoteca>
 <CD>
  <TITULO>Perfil</TITULO>
  <FAIXA.TITULO>Nossa
Canção</FAIXA.TITULO>
  <FAIXA.COMPOSITORES>
   <TP_COMPOSITOR>
    <NOME>Roberto
Carlos</NOME>

</TP_COMPOSITOR>
   <TP_COMPOSITOR>
    <NOME>Erasmo Carlos</NOME>
   </TP_COMPOSITOR>
```

Outras funcionalidades

- Resultado (Cont.)

```
</FAIXA.COMPOSITORES>
 </CD>
 <CD>
  <TITULO>Perfil</TITULO>
  <FAIXA.TITULO>Negro
Gato</FAIXA.TITULO>
  <FAIXA.COMPOSITORES>
   <TP_COMPOSITOR>
    <NOME>Roberto
Carlos</NOME>
</TP_COMPOSITOR>
   <TP_COMPOSITOR>
    <NOME>Erasmo Carlos</NOME>
   </TP_COMPOSITOR>
```


Outras funcionalidades

- Resultado (Cont.)

```
<TP_COMPOSITOR>
   <NOME>Rossini Pinto</NOME>
  </TP_COMPOSITOR>
 </FAIXA.COMPOSITORES>
 </CD>
<CD>
 <TITULO>Páginas da Vida</TITULO>
 <FAIXA.TITULO>Copacabana</FAIXA.TITULO>
<FAIXA.COMPOSITORES>
 <TP_COMPOSITOR>
  <NOME>Antonio Maria</NOME>
 </TP_COMPOSITOR>
</FAIXA.COMPOSITORES>
```


Outras funcionalidades

- Resultado (Cont.)

```
<TP_COMPOSITOR>
   <NOME>Rossini Pinto</NOME>
  </TP_COMPOSITOR>
 </FAIXA.COMPOSITORES>
 </CD>
<CD>
 <TITULO>Páginas da Vida</TITULO>
 <FAIXA.TITULO>Copacabana</FAIXA.TITULO>
<FAIXA.COMPOSITORES>
 <TP_COMPOSITOR>
  <NOME>Antonio Maria</NOME>
 </TP_COMPOSITOR>
</FAIXA.COMPOSITORES>
```


Outras funcionalidades

- Resultado (Cont.)

```
</CD>
 <CD>
  <TITULO>Páginas da Vida</TITULO>

<FAIXA.TITULO>Desafinado</FAIXA.TITULO>
 <FAIXA.COMPOSITORES>
  <TP_COMPOSITOR>
   <NOME>João
Gilberto</NOME>
  </TP_COMPOSITOR>
 </FAIXA.COMPOSITORES>
 </CD>
 <CD>
```


Outras funcionalidades

- Resultado (Cont.)

```
<TITULO>Belíssima</TITULO>
 <FAIXA.TITULO>Todo mundo gosta de mim</FAIXA.TITULO>

<FAIXA.COMPOSITORES>
  <TP_COMPOSITOR>
   <NOME>Ronaldo Boscoli</NOME>
  </TP_COMPOSITOR>

<TP_COMPOSITOR>
   <NOME>Nelson Mota</NOME>
  </TP_COMPOSITOR>
```


Outras funcionalidades

- Resultado (Cont.)

```
 <TP_COMPOSITOR>
   <NOME>Tom
Zé</NOME>
  </TP_COMPOSITOR>
 </FAIXA.COMPOSITORES>
 </CD>
</Discoteca>
```

Exercício Extra XML Opcional

- Gerar arquivo XML a partir de consulta em uma das tabelas do modelo OR implementado pela equipe no Oracle
  - Uma consulta diferente para cada membro da equipe
- Criar uma DTD ou XML Schema apropriado para o arquivo XML gerado
  - 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

Vale até 1,0 ponto na prova para passar por média ou ir para a final