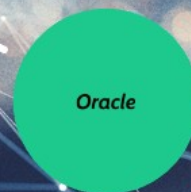





XML  
com  
Oracle



***Gustavo Gatto Scarano***

Analista/Programador de Sistema desde 2008  
Desenvolvedor Oracle PL/SQL desde 2009

 contato.ggs@gmail.com

 (19) 98334-7853

 gugatto



## ***XML (Extensible Markup Language)***

O QUE É? É uma recomendação da W3C para gerar linguagens de marcação para necessidades especiais.

Seu propósito principal é a facilidade de compartilhamento de informações por intermédio da internet.

```
<?xml version="1.0"?>
<quiz>
  <questão>
    Qual o segundo nome de
    Sofia Amundsen?
  </questão>
  <resposta>
    Amundsen
  </resposta>
  <!-- Nota: Precisamos
    perguntar mais.-->
</quiz>
```

**XML**

***Propósitos***

***Por que XML?***

**Armazenamento de informação:** XML é portátil e não é proprietário, pode ser usado para armazenamento de informação em qualquer plataforma, por ter um padrão internacional;

**Mensagens e transferência de dados:** XML também é extremamente usado para juntar ou encapsular informação para ser trocada entre diferentes sistemas computacionais que seriam incapazes de se comunicar, sem precisar se preocupar com tipo de dado;

**Web Services:** o XML é usado para gestão da informação e transmissão.

entre outros...



## **POR QUE XML?**

- É recomendado pelo W3C (World Wide Web Consortium);
- É padrão aberto, você não precisa pagar nada para usar;
- Existem várias ferramentas e editores bons no mercado, e o melhor: free;
- Simplicidade e Legibilidade, tanto para humanos quanto para computadores;
- Separação do conteúdo da formatação;
- Concentração na estrutura da informação, e não na sua aparência;
- Possibilidade de criar sua própria sintaxe de dados, ou seja, estruturar os dados da forma que achar melhor, através da criação ilimitada de tags;
- Possui suporte a Unicode, permitindo que praticamente qualquer informação escrita em língua humana possa ser transmitida;
- Permite validação, o que torna os teste unitários mais efetivos, e a construção de aplicações bem mais fácil;
- Permite criar um padrão de arquivo através do XSD.



## **POR QUE ORACLE?**

A partir da versão 9i, o Oracle apresenta um novo tipo de objeto, que é o XMLType.

Principalmente por este motivo, o Oracle, a partir da versão 9i, pode ser considerado um SGBD com forte suporte a XML.

*e o que significa  
isso?*

- Armazenar e recuperar XML como um todo no banco de dados;
- Fazer consultas em elementos XML;
- Não se precisa atualizar o documento em pedaços e sim o documento inteiro; e
- Preparar para futuras otimizações. Devido ao fato de que toda funcionalidade relacionada a XML somente tem suporte pelo tipo XMLType. Qualquer futura otimização poderá ser automaticamente aproveitada sem a necessidade de reescrever a aplicação.

*Para desenvolvimento de uma aplicação com banco de dados XML é preciso saber quais as funções que auxiliam a manipulação de dados XMLType no Oracle.*

Funções

Carregando a partir de diretório

Exemplo de XML

Como consultar um XML

Como consultar múltiplos nós

Como criar um XML



Função	Parâmetro	Retorno	Descrição
createXML ()	xmlval: VARCHAR2	XMLType	Cria uma instância de um XMLType a partir de uma String.
createXML ()	xmlval: CLOB	XMLType	Cria uma instância de um XMLType a partir de um CLOB.
existsNode ()	xpath: VARCHAR2	NUMBER	Dada uma expressão XPath, verifica se o XPath aplicado ao documento pode retornar nós válidos.
extract()	xpath: VARCHAR2	XMLType	Dada uma expressão XPath, aplica o XPath ao documento e retorna o fragmento como XMLType.
isFragment ()	-	NUMBER	Verifica se o documento é realmente um fragmento. Retorna 1 se a instância XML contém um fragmento e 0 caso contrário.
getClobVal ()	-	CLOB	Retorna o documento como um CLOB.
getString Val()	-	VARCHAR 2	Retorna o valor XML como uma String.
getNumber Val()	-	NUMBER	Retorna o valor numérico para o qual o XMLType aponta.

## Carregando a partir de diretório:

```
PROCEDURE recebeXML
(
  diretorio,
  nome_arquivo
) IS
  variavel_clob CLOB;
BEGIN
  -- Carregando o XML em uma variavel.
  SELECT xmlserialize(document
    xmltype(bfilename(diretorio, nome_arquivo), nls_charset_id('AL32UTF8'))) AS xmldoc
  INTO variavel_clob
  FROM dual;
END;
--
INSERT INTO tabela_xml
(primary_key_tabela, -- NUMBER
arquivo_xml) -- XMLTYPE
VALUES
(I,
xmltype.createxml(variavel_clob));
--
END recebeXML;
```

```

<?xml version="1.0" encoding="UTF-8"?>
<PurchaseOrders>
  <PurchaseOrder>
    <PurchaseOrderNumber>
      <OrderNumber>99503</OrderNumber>
    </PurchaseOrderNumber>
    <Address>
      <Type>Shipping</Type>
      <Name>Ellen Adams</Name>
      <Street>123 Maple Street</Street>
      <City>Mill Valley</City>
      <State>CA</State>
      <Zip>10999</Zip>
      <Country>USA</Country>
    </Address>
    <Address>
      <Type>Billing</Type>
      <Name>Tai Yee</Name>
      <Street>8 Oak Avenue</Street>
      <City>Old Town</City>
      <State>PA</State>
      <Zip>95819</Zip>
      <Country>USA</Country>
    </Address>
    <DeliveryNotes>Please leave packages in shed by driveway.</DeliveryNotes>
    <Items>
      <Item>
        <PartNumber>872-AA</PartNumber>
        <ProductName>Lawnmower</ProductName>
        <Quantity>1</Quantity>
        <USPrice>148.95</USPrice>
        <Comment>Confirm this is electric</Comment>
      </Item>
    </Items>
  </PurchaseOrder>
</PurchaseOrders>

```

## Exemplo de XML

O arquivo está disponível no repositório do  
meetup no github  
(<https://github.com/plsqlcamp/Meetup>)



## Como consultar um XML

SQL Output Statistics

```
1 SELECT tabela_xml.primary_key_tabela,
2     xml_arq_1.*
3 FROM   tabela_xml,
4     xmltable(xmlnamespaces('http://www.w3.org/2001/XMLSchema-instance' AS "schema"),
5              '/PurchaseOrders/PurchaseOrder'
6              passing tabela_xml.arquivo_xml
7              COLUMNS
8              PurchaseOrderNumber VARCHAR(500) PATH 'PurchaseOrderNumber/OrderNumber',
9              Address XMLTYPE PATH 'Address'
10             xml_arq_1
11
```

PRIMARY\_KEY\_TABELA | PURCHASEORDERNUMBER | ADDRESS

1	1 99503	... <XMLTYPE> ...
2	1 88888	... <XMLTYPE> ...

<Address>  
<Type>Shipping</Type>  
<Name>Ellen Adams</Name>  
<Street>123 Maple Street</Street>  
<City>Mill Valley</City>  
<State>CA</State>  
<Zip>10999</Zip>  
<Country>USA</Country>  
</Address>  
<Address>  
<Type>Billing</Type>  
<Name>Tai Yee</Name>  
<Street>8 Oak Avenue</Street>  
<City>Old Town</City>  
<State>PA</State>  
<Zip>95819</Zip>  
<Country>USA</Country>  
</Address>

Comando para criar uma estrutura XML (permitindo trabalhar com as tag dentro do XML)

Colunas que serão extraídas do XML para formato de tabela.

Origem/Campo XMLTYPE da tabela com o XML original

Subtipo para criar outra tabela XMLTYPE contendo sub-nós do XML

## Como consultar múltiplos nós

Passando XML original para extração de nós pais.

```
SQL Output Statistics
1 SELECT tabela_xml.primary_key_tabela,
2     xml_arq_1.*,
3     xml_arq_2.*
4 FROM   tabela_xml,
5     xmltable(xmlnamespaces('http://www.w3.org/2001/XMLSchema-instance' AS "schema"),
6         '/PurchaseOrders/PurchaseOrder'
7     ) passing tabela_xml.arquivo_xml
8     COLUMNS
9         PurchaseOrderNumber VARCHAR(500) PATH 'PurchaseOrderNumber/OrderNumber',
10        Address XMLTYPE PATH 'Address'
11    ) xml_arq_1,
12    xmltable(xmlnamespaces('http://www.w3.org/2001/XMLSchema-instance' AS "schema"),
13        '/Address'
14    ) passing xml_arq_1.Address
15    columns
16        Tipo VARCHAR(500) PATH 'Type',
17        Ender VARCHAR(500) PATH 'Name',
18        Street VARCHAR(500) PATH 'Street',
19        State VARCHAR(500) PATH 'State',
20        Zip VARCHAR(500) PATH 'Zip',
21        Country VARCHAR(500) PATH 'Country' ) xml_arq_2
22 WHERE  tabela_xml.primary_key_tabela = 1;
23
```

Extraindo informações de nós filhos

	PRIMARY_KEY_TABELA	PURCHASEORDERNUMBER	ADDRESS	TIPO	ENDER	STREET	STATE	ZIP	COUNTRY
1	1	99503	<XMLTYPE>	Shipping	Ellen Adams	123 Maple Street	CA	10999	USA
2	1	99503	<XMLTYPE>	Billing	Tai Yee	8 Oak Avenue	PA	95819	USA
3	1	88888	<XMLTYPE>	Shipping	Teste Adams	123 Mill Valley Street	TO	11111	USA
4	1	88888	<XMLTYPE>	Billing	Tai TEste	8 Old Town Avenue	RO	88888	USA

SQL Output Statistics

```

18 SELECT xmlelement("orders",
19       xmlagg(xmlelement("order",
20       xmlforest(d.ordernumber AS "ordernumber",
21       (SELECT xmlagg(xmlelement("endereco",
22       xmlforest(e.ender AS "ender",
23       e.street AS "street",
24       e.state AS "state",
25       e.zip AS "zip",
26       e.country AS "country"))))
27 FROM   tabela e
28 WHERE  e.ordernumber = d.ordernumber) "emp_list")))) AS "depts"
29 FROM   tabela d
30 WHERE  d.ordernumber = 99503
31 AND ROWNUM = 1;
32

```

**XMLElement** - Permite a criação de um elemento xml (Ex: XMLEMENT("Name", e.job\_id ' e.last\_name)  
Resultado:

<Name>MK\_MAN Hartstein</Name>

**XMLAgg** - Permite trabalhar com loop dentro do XML

**XMLForest** - permite adição de mais de um parametro (caso não use "Alias" a TAG é criada com o nome do campo da tabela)

Ex: XMLFOREST(e.employee\_id, e.last\_name, e.salary)

<EMPLOYEE\_ID>204</EMPLOYEE\_ID>

<LAST\_NAME>Baer</LAST\_NAME>

<SALARY>10000</SALARY>

Todos os arquivos e comandos estão disponibilizados no repositório Github (<https://github.com/plsqlcamp/Meetup>)

Como gerar um XML

```

<?xml version="1.0" encoding="UTF-8"?>
<orders>
  <order>
    <ordernumber>99503</ordernumber>
    <emp_list>
      <endereco>
        <ender>Ellen Adams</ender>
        <street>123 Maple Street</street>
        <state>CA</state>
        <zip>10999</zip>
        <country>USA</country>
      </endereco>
      <endereco>
        <ender>Tai Yee</ender>
        <street>8 Oak Avenue</street>
        <state>PA</state>
        <zip>95819</zip>
        <country>USA</country>
      </endereco>
    </emp_list>
  </order>
</orders>

```