XML a ORACLE

Petr Davídek

Jak to vypadalo dříve (verze 8)

Nebyla „nativní“ podpora XML, nutné doinstalovat (java-based)

- **XML ukládáno jako LOB, BFILE**
- DBMS_XMLQuery, XMLDOM, XML_PARSER – základní balíky pro práci s XML
- sql skript $ORACLE_HOME/rdbms/admin/catxsu.sql)
- loadjava -v -r -grant PUBLIC $ORACLE_HOME/xdk/lib/xmlparserv2.jar
- $ORACLE_HOME/xdk/plsql/parser/bin/load.sql
- Granty a synonyma
  - grant execute on xmldom to public;
  - grant execute on xmlparser to public;
  - create public synonym xmldom for sys.xmldom;
  - create public synonym xmlparser for sys.xmlparser;

Práce s XML ve verzi 8

výsledek SQL dotazu do XML 1

```
DECLARE
Ctx    DBMS_XMLQuery.ctxType;   -- SQL do XML
xml    clob;
emp_no NUMBER := 7369;
xmlc   varchar2(4000);
off    integer := 1;
len    integer := 4000;
BEGIN
  Ctx := DBMS_XMLQuery.newContext('SELECT * FROM emp WHERE empno = :empno');
  DBMS_XMLQuery.setBindValue(Ctx, 'empno', emp_no);
  xml := DBMS_XMLQuery.getXML(Ctx);
  DBMS_XMLQuery.closeContext(Ctx);
  DBMS_LOB.READ(xml, len, off, xmlc);
  DBMS_OUTPUT.PUT_LINE(xmlc);
END;
/
```

Práce s XML ve verzi 8

výsledek SQL dotazu do XML 2

```
<ROWSET>
  <ROW num="1">
    <EMPNO>7369</EMPNO>

<ENAME>SMITH</ENAME>
    <JOB>CLERK</JOB>
    <MGR>7902</MGR>

<HIREDATE>12/17/1980 0:0:0</HIREDATE>
    <SAL>800</SAL>

<DEPTNO>20</DEPTNO>
  </ROW>
</ROWSET>
```

Práce s XML ve verzi 8

Parsing XML dokumentu

- Balík XMLParser

```
DECLARE
 p_parser    xmlparser.parser,
 l_doc       xmldom.DOMDocument;
 l_xml       CLOB;

BEGIN
  SELECT clob_content INTO l_xml FROM nestt_docs WHERE oid = p_XML_oid;

  xmlparser.setValidationMode(p_parser, false);
  xmlparser.ParseClob(p_parser, l_pom_clob);
  l_doc := xmlparser.getDocument(p_parser);
END;
/
```

- Další možnosti parsingu
  - ParseFile
  - ParseUrl

Práce s XML ve verzi 8

Zpracování XML pomocí DOM

```
DECLARE
 p_doc              xmldom.DOMDocument,
 p_nodeName  varchar2(20) := 'EMP';
 l_listData       xmldom.DOMNodeList;
 l_nodeData     xmldom.DOMNode;

BEGIN
  l_listData := xmldom.getElementsByTagName(p_doc, p_NodeName);
  l_nodeData := xmldom.item(l_listData, 0);
  xmldom.writeToBuffer(l_nodeData, l_value);
END;
/
```

- Další funkce pro parsing a vytváření XML v balíku XMLDOM

XML v ORACLE (9i, 10g)

- Od verze 9.2.0 podpora XML – XMLTYPE – nový datový typ
- ORACLE XMLDB – nejedná se o xml nativní DB (jak tvrdí ORACLE), ale o tzv. xml-enabled DB.
- Způsoby uložení XML
  - LOB, BFILE
  - Obecný XML dokument (ukládán jako LOB)
  - XML dokument odpovídající schématu (DTD, XML Schema) – zde je využito mapování fragmentů XML dokumentu do relačního modelu. Při vytváření XMLTYPE sloupečku je zároveň vytvořeno schéma a při vkládání XML do tohoto sloupečku je proveden parsing dokumentu a jeho části jsou uloženy do takto vzniklého modelu. Naopak při dotazech (XPATH) není prohledáván celý LOB ale je opět využito tohoto modelu.
    - Výhody – možnost indexování, rychlejší vyhledávání
    - Nevýhody – jistá režije spjatá s mapováním XML, při dotazech vracejících fragmenty XML je nutná jejich rekonstrukce z relačního modelu, nesnadná změna XML Schématu

XMLTYPE

- Interní typ (object, TYPE) ORACLE
- Může být použit jako typ tabulky, sloupečku, jako datový typ v PL/SQL, při definici návratových hodnot, při definici sloupečků ve view

```
CREATE TABLE orders OF XMLTYPE;

CREATE TABLE orders2 (id number, order XMLTYPE);

declare xml XMLTYPE;
begin
 xml := xmltype('<?xml><book>...</book>');
end;
/
```

XMLTYPE
XMLTYPE s daným schématem 1

- Pro ukládání XML dat, které odpovídají xml schematu je možno vytvořit XMLTYPE sloupeček na základě tohoto schématu. Každý vkládaný dokument je nejprve zvalidován a teprve potom může být uložen do sloupečku
- XML schéma je nutno nejprve zaregistrovat
- Nutný grant - XDBADMIN

PROCEDURE DBMS_XMLSCHEMA.REGISTERSCHEMA

Argument Name	Type	In/Out	Default?
SCHEMAURL	VARCHAR2		
SCHEMADOC	URITYPE		
LOCAL	BOOLEAN		
GENTYPES	BOOLEAN		
GENBEAN	BOOLEAN		
GENTABLES	BOOLEAN		
FORCE	BOOLEAN		
OWNER	VARCHAR2		
ENABLEHIERARCHY	BINARY_INTEGER		
OPTIONS	BINARY_INTEGER		

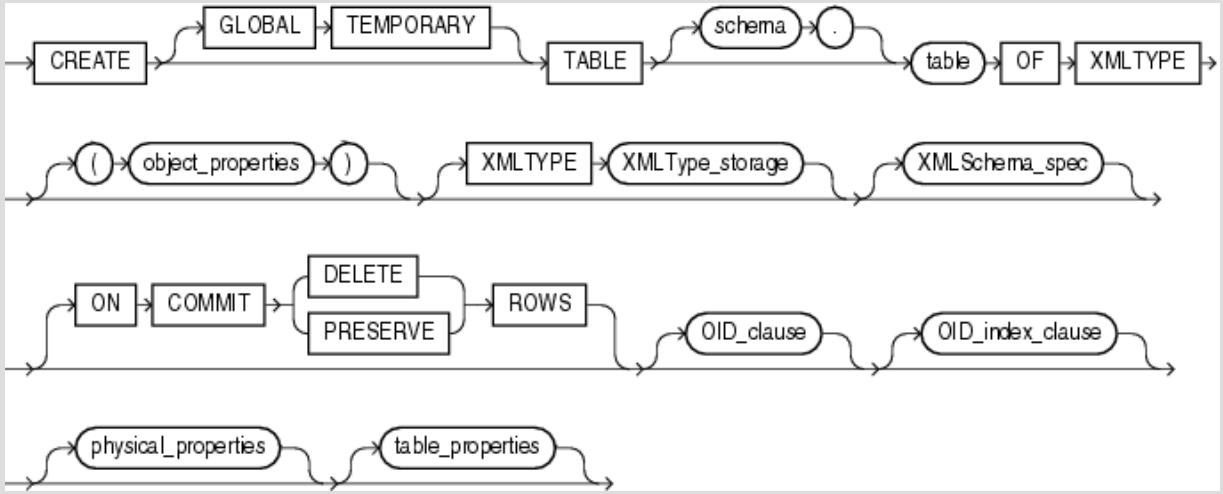
XMLTYPE
XMLTYPE s daným schématem 2

```
declare
  l_doc varchar2(4000) := '<?xml version="1.0" encoding="UTF-8"?>
<schema targetNamespace="http://akar.gtsgroup.cz/Durian/Phone/Statement/Statement"
 xmlns="http://www.w3.org/2001/XMLSchema">
 <element name="statement">
  <complexType>
   <sequence>
    <element name="header">
     <complexType>
      <sequence>
       <element maxOccurs="unbounded" ref="dpsr:request"/>
      </sequence>
     </complexType>
    </element>
</schema>';
begin
  DBMS_XMLSCHEMA.registerSchema(
    SCHEMAURL => 'http://akar.gtsgroup.cz/Durian/Phone/Statement/Statement',
       SCHEMADOC => l_doc,
       GENTABLES => FALSE
       );
end;
/
```

```
CREATE TABLE request (id NUMBER not null, xml_request XMLType not null)  XMLType COLUMN xml_request XMLSCHEMA
"http://akar.gtsgroup.cz/Durian/Phone/Statement/Statement" ELEMENT "statement";
/
```

- Element se odkazuje na rootovský element XML dokumentu
- XMLSCHEMA se odkazuje na zaregistrované XML schéma

XMLTYPE
XMLTYPE s daným schématem 3




```
CREATE TABLE purchaseorder_as_column (
  id NUMBER,
  xml_document XMLType,
  UNIQUE (xml_document."XMLDATA"."Reference"),
  FOREIGN KEY (xml_document."XMLDATA"."User") REFERENCES hr.employees (email))
  XMLTYPE COLUMN xml_document
  XMLSCHEMA "http://xmlns.oracle.com/xdb/documentation/purchaseOrder.xsd"
  ELEMENT "PurchaseOrder"
  VARRAY xml_document."XMLDATA"."Actions"."Action"
    STORE AS TABLE action_table2
                  ((PRIMARY KEY (NESTED_TABLE_ID, SYS_NC_ARRAY_INDEX$))
                  ORGANIZATION INDEX OVERFLOW)
  VARRAY xml_document."XMLDATA"."LineItems"."LineItem"
    STORE AS TABLE lineitem_table2
                  ((PRIMARY KEY (NESTED_TABLE_ID, SYS_NC_ARRAY_INDEX$))
                  ORGANIZATION INDEX OVERFLOW)
  LOB (xml_document."XMLDATA"."Notes")
    STORE AS (TABLESPACE USERS ENABLE STORAGE IN ROW
               STORAGE(INITIAL 4K NEXT 32K));
```

XMLTYPE
XMLTYPE s daným schématem 4

- Na zrušení zaregistrovaného schématu slouží

  DBMS_XMLSCHEMA.deleteSchema

- Na změnu zaregistrovaného schématu slouží

  DBMS_XMLSCHEMA.copyEvolve

SQL fce pro manipulaci s XML 1
XMLTYPE, getXxxVal

- XMLTYPE - vložení XML dokumentu do XMLTYPE sloupečku
- Nejčastěji pomocí metody XMLTYPE, která převádí varchar2, CLOB na XMLTYPE

insert into xmltable(id, xmldoc) values (1, XMLTYPE('<?xml>.....'));

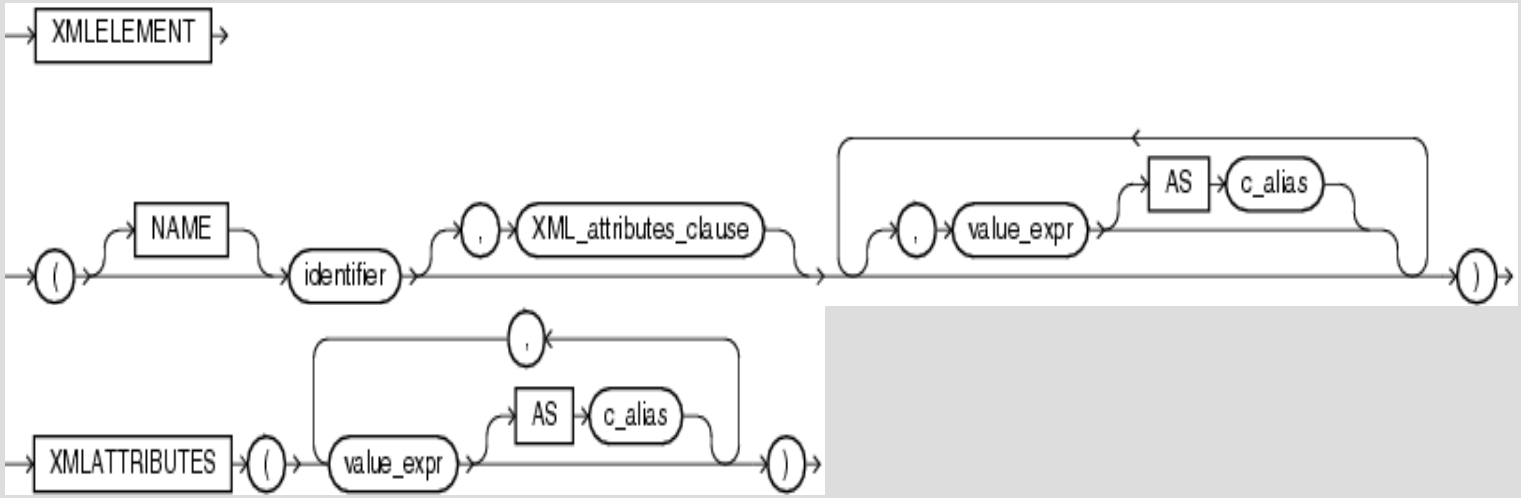
- GetClobVal() - získání znakové (číselné) hodnoty z XMLTYPE (XML strom převede na zobrazitelnou formu)
- GetClobVal(), GetStringVal(), getNumberVal()

select xmldoc.getClobVal() from xml_docs where id=12;

```
RESULT
-----------------------------------
<EMPLOYEES>
  <EMP>
     <EMPNO>112</EMPNO>
     <EMPNAME>Joe</EMPNAME>
     <SALARY>50000</SALARY>
  </EMP>
  <EMP>
     <EMPNO>217</EMPNO>
     <EMPNAME>Jane</EMPNAME>
     <SALARY>60000</SALARY>
  </EMP>
</EMPLOYEES>
```

SQL fce pro manipulaci s XML 2
XMLElement, XMLAttributes

- Slouží k vytvoření fragmentu XML na základě vybraných dat




```
SELECT XMLElement("Emp", XMLAttributes(
                         e.employee_id as "ID",
                        e.first_name ||' ' || e.last_name AS "name"))
  AS "RESULT"
  FROM hr.employees e
  WHERE employee_id > 200;


RESULT
------------------------------------------------
<Emp ID="201" name="Michael Hartstein"></Emp>
<Emp ID="202" name="Pat Fay"></Emp>
<Emp ID="203" name="Susan Mavris"></Emp>
<Emp ID="204" name="Hermann Baer"></Emp>
<Emp ID="205" name="Shelley Higgins"></Emp>
<Emp ID="206" name="William Gietz"></Emp>
```

SQL fce pro manipulaci s XML 3
XMLForest

- Podobné jako XMLElement, umožňuje vybrat najednou více elemetů (les)


```
SELECT XMLElement("Emp",
                XMLAttributes(e.first_name ||' '|| e.last_name AS "name"),
                XMLForest(e.hire_date, e.department AS "department"))
AS "RESULT"
FROM employees e WHERE e.department_id = 20;
```

```
 RESULT
 ------------------------------------
 <Emp name="Michael Hartstein">
   <HIRE_DATE>1996-02-17</HIRE_DATE>
   <department>20</department>
 </Emp>
 <Emp name="Pat Fay">
   <HIRE_DATE>1997-08-17</HIRE_DATE>
   <department>20</department>
 </Emp>
```

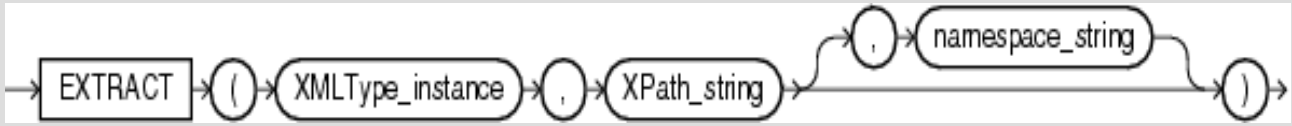
SQL fce pro manipulaci s XML 4
XMLAGG

- „agreguje“ (slučuje) více xml fragmentů do jednoho, umožňuje budovat strukturované XML dokumenty ze selektů

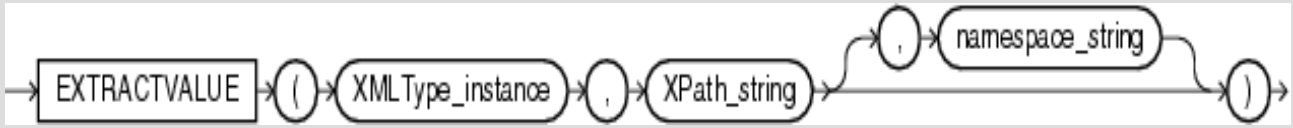

```
SELECT XMLElement("Department", XMLAgg(XMLElement("Employee",
                                                  e.job_id||' '||e.last_name)
                                       ORDER BY e.last_name))
  AS "Dept_list"
  FROM hr.employees e
  WHERE e.department_id = 30 OR e.department_id = 40;


Dept_list
------------------
<Department>
  <Employee>PU_CLERK Baida</Employee>
  <Employee>PU_CLERK Colmenares</Employee>
  <Employee>PU_CLERK Himuro</Employee>
  <Employee>PU_CLERK Khoo</Employee>
  <Employee>HR_REP Mavris</Employee>
  <Employee>PU_MAN Raphaely</Employee>
  <Employee>PU_CLERK Tobias</Employee>
</Department>
```

SQL fce pro manipulaci s XML 5
extract, extractValue, existsNode

- Extract – vrací fragment XML dokumentu, který odpovídá XPATH výrazu



- ExtractValue – podobně jako Extract, ale nevrací fragment ale maximálně jednu hodnotu, pokud XPATH výrazu odpovídá více elekemtů, atributů, je ohlášena chyba




```
SELECT extract(OBJECT_VALUE, '/PurchaseOrder/Reference') "REFERENCE"    FROM purchaseorder

REFERENCE
--------------------------------------------------------------
<Reference>AMCEWEN-20021009123336271PDT</Reference>
<Reference>SKING-20021009123336321PDT</Reference>


SELECT extractValue(OBJECT_VALUE, '/PurchaseOrder/Reference') "REFERENCE"

REFERENCE
--------------------------------------------------------------
AMCEWEN-20021009123336271PDT
20021009123336321PDT
```

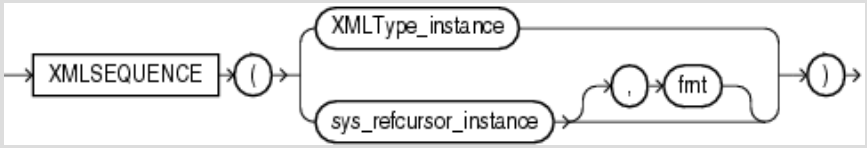
SQL fce pro manipulaci s XML 6
ExistsNode

- ExistsNode – vrací 1, pokud v dokumentu existuje fragment, odpovídající XPATH výrazu


```
SELECT OBJECT_VALUE
  FROM purchaseorder
  WHERE existsNode(OBJECT_VALUE, '/PurchaseOrder[SpecialInstructions="Expedite"]')
        = 1;

OBJECT_VALUE
--------------------------------------------------------------------------------
<PurchaseOrder xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
<PurchaseOrder xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
<PurchaseOrder xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
<PurchaseOrder xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
<PurchaseOrder xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
<PurchaseOrder xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
```

SQL fce pro manipulaci s XML 7
XMLSequence

- XMLSequence – vrací colectionu XMLTYPů (pole proměnné velikosti). Nejčastější použití je společně s table. Operátor table z collection udělá něco jako virtuální tabulku, do které se můžeme odkazovat dalšími selekty

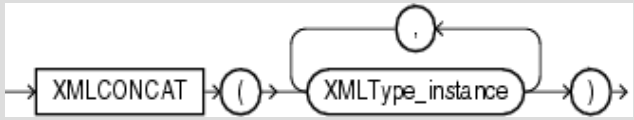



```
SELECT value(T).getstringval() Attribute_Value
  FROM table(XMLSequence(extract(XMLType('<A><B>V1</B><B>V2</B><B>V3</B></A>'),
                                 '/A/B'))) T;
ATTRIBUTE_VALUE
----------------------
<B>V1</B>
<B>V2</B>
<B>V3</B>
```

```
SELECT value(em).getClobVal() AS "XMLTYPE" FROM table(XMLSequence(Cursor(SELECT * FROM hr.employees
WHERE employee_id = 104))) em;
XMLTYPE
--------------------------------------------------
 <ROW>
  <EMPLOYEE_ID>104</EMPLOYEE_ID>
  <FIRST_NAME>Bruce</FIRST_NAME>
  <LAST_NAME>Ernst</LAST_NAME>
  <EMAIL>BERNST</EMAIL>
  <HIRE_DATE>21-MAY-91</HIRE_DATE>
  <SALARY>6000</SALARY>
  <MANAGER_ID>103</MANAGER_ID>
  <DEPARTMENT_ID>60</DEPARTMENT_ID>
 </ROW>
```

SQL fce pro manipulaci s XML 8
XMLConcat

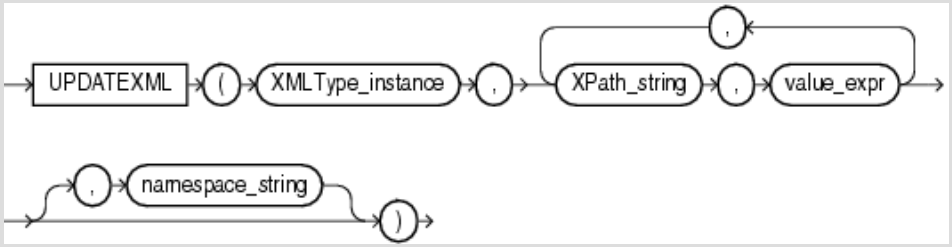
- Fce s proměnným počtem argumentů, výsledkem je XML fragment, řetězící jednotlivé argumenty




```
SELECT XMLConcat(XMLSequenceType(
                  XMLType('<PartNo>1236</PartNo>'),
                  XMLType('<PartName>Widget</PartName>'),
                  XMLType('<PartPrice>29.99</PartPrice>'))).getClobVal()
  AS "RESULT"
  FROM DUAL;

 RESULT
 ---------------
 <PartNo>1236</PartNo><PartName>Widget</PartName><PartPrice>29.99</PartPrice>


 SELECT XMLConcat(XMLElement("first", e.first_name),
                  XMLElement("last", e.last_name))
   AS "RESULT"
   FROM employees e;


 RESULT
 ------------------------------------------
 <first>Den</first><last>Raphaely</last>
 <first>Alexander</first><last>Khoo</last>
 <first>Shelli</first><last>Baida</last>
 <first>Sigal</first><last>Tobias</last>
 <first>Guy</first><last>Himuro</last>
 <first>Karen</first><last>Colmenares</last>
```

SQL fce pro manipulaci s XML 9
UPDATEXML

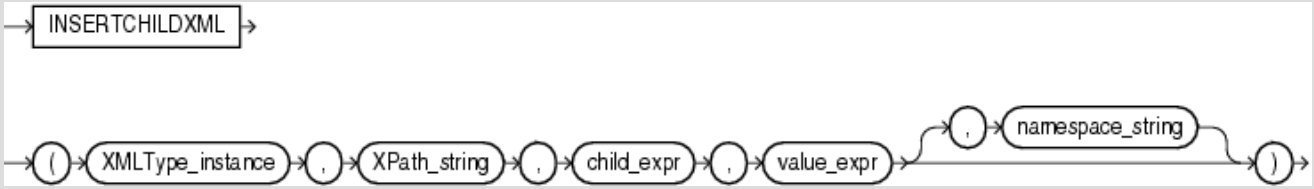
- Nahradí všechny části XML dokumentu, které odpovídají XPATH výrazu




```
SELECT extract(OBJECT_VALUE, '/PurchaseOrder/Actions/Action[1]') ACTION FROM purchaseorder
  WHERE existsNode(OBJECT_VALUE, '/PurchaseOrder[Reference="SBELL-20021009123336O1PDT"]') = 1;

ACTION
--------------------------------
<Action>
  <User>SVOLLMAN</User>
</Action>

UPDATE purchaseorder
  SET OBJECT_VALUE = updateXML(OBJECT_VALUE, '/PurchaseOrder/Actions/Action[1]/User/text()', 'SKING')
  WHERE existsNode(OBJECT_VALUE, '/PurchaseOrder[Reference="SBELL-20021009123336O1PDT"]') = 1;

SELECT extract(OBJECT_VALUE, '/PurchaseOrder/Actions/Action[1]') ACTION
  FROM purchaseorder
  WHERE existsNode(OBJECT_VALUE,'/PurchaseOrder[Reference="SBELL-20021009123336O1PDT"]') = 1;

ACTION
--------------------------------
<Action>
  <User>SKING</User>
</Action>
```

SQL fce pro manipulaci s XML 10
INSERTCHILDXMLL

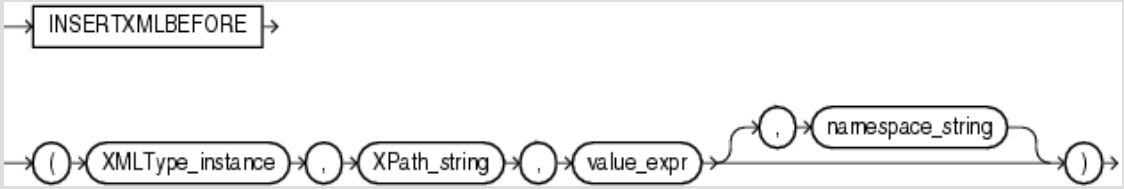
- Vkládá nový element(y) či nový atribut pod otcovský element




```
SELECT extract(OBJECT_VALUE,  '/PurchaseOrder/LineItems/LineItem[@ItemNumber="222"]')  CHLD FROM purchaseorder
  WHERE existsNode(OBJECT_VALUE,  '/PurchaseOrder[Reference="AMCEWEN-20021009123336171PDT"]')  = 1;

CHLD
--------------------------------------------------------------------------------

UPDATE purchaseorder  SET OBJECT_VALUE = insertChildXML(OBJECT_VALUE,
                    '/PurchaseOrder/LineItems',
                    'LineItem',
                    XMLType('<LineItem ItemNumber="222">
                              <Description>The Harder They Come</Description>
                              <Part Id="953562951413"
                                  UnitPrice="22.95"
                                  Quantity="1"/>
                             </LineItem>'))
  WHERE existsNode(OBJECT_VALUE,  '/PurchaseOrder[Reference="AMCEWEN-20021009123336171PDT"]')  = 1;

SELECT extract(OBJECT_VALUE,  '/PurchaseOrder/LineItems/LineItem[@ItemNumber="222"]')  CHLD FROM purchaseorder
  WHERE existsNode(OBJECT_VALUE,  '/PurchaseOrder[Reference="AMCEWEN-20021009123336171PDT"]')  = 1;

CHLD
--------------------------------------------------------------------------------
<LineItem ItemNumber="222">
  <Description>The Harder They Come</Description>
  <Part Id="953562951413" UnitPrice="22.95" Quantity="1"/>
</LineItem>
```

SQL fce pro manipulaci s XML 11
INSERTXMLBEFORE

- Vkládá nový element bezprostředně před element daný XPATH
- Výsledkem XPATH dotazu nesmí být atribut




```
SELECT extract(OBJECT_VALUE, '/PurchaseOrder/LineItems/LineItem[1]') CHLDS FROM purchaseorder
  WHERE existsNode(OBJECT_VALUE, '/PurchaseOrder[Reference="AMCEWEN-20021009123336171PDT"]') = 1;

CHLDS
--------------------------------------------------------------
<LineItem ItemNumber="1"><Description>Salesman</Description><Part Id="37429158920" UnitPrice="39.95" Quantity="2"/></LineItem>

UPDATE purchaseorder SET OBJECT_VALUE = insertXMLbefore(OBJECT_VALUE, '/PurchaseOrder/LineItems/LineItem[1]'
                        ,XMLType('<LineItem ItemNumber="314">
                          <Description>Brazil</Description>
                          <Part Id="314159265359"
                              UnitPrice="69.95"
                              Quantity="2"/>
                        </LineItem>'))
  WHERE existsNode(OBJECT_VALUE,'/PurchaseOrder[Reference="AMCEWEN-20021009123336171PDT"]')= 1;

SELECT extract(OBJECT_VALUE, '/PurchaseOrder/LineItems/LineItem[position() <= 2]') CHLDS FROM purchaseorder WHERE
existsNode(OBJECT_VALUE, '/PurchaseOrder[Reference="AMCEWEN-00210091233336171PDT"]')= 1;

CHLDS
--------------------------------------------------------------------------------
<LineItem ItemNumber="314"><Description>Brazil</Description><Part Id="314159265359" UnitPrice="69.95" Quantity="2"/></LineItem>
<LineItem ItemNumber="1"><Description>Salesman</Description><Part Id="37429158920" UnitPrice="39.95" Quantity="2"/></LineItem>
```

SQL fce pro manipulaci s XML 10

- XMLPI – pro generovaní PI (process instrunctions)
- XMLComment – generování komentářů
- XMLCDATA – generování sekcí CDATA
- XMLROOT – generování XML „hlavičky“ <?xml version=“1.0“>
- XMLCOLATTVAL – podobné jako XMLFOREST, elementy jsou brány jako hodnoty atributů, není třeba escapovat
- APPENDCHILDXML - vkládá fragment jako poslední dítě k elementu dle XPATH
- DELETEXML – smaže cokoliv, co odpovídá XPATH

Indexování XMLTYPE

- Jednoduchý index založený na extractValue – XPATH výraz je přeložen na extractValue, málo účinné, možnýpouze tehdy, pokud hledaný XPATH je max. 1X v dokumentu

```
CREATE INDEX ipurchaseorder_rejectedby ON purchaseorder (extractValue(OBJECT_VALUE, '/PurchaseOrder/Reject/User'));
```

- B-tree index – jako jednoduchý, místo extractValue se použije Extract
- Funkční index – používá se pro XMLTYPE uložený v CLOBu. Při manipulaci je zavolána příslušná fce (např. ExtractValue) – může se tím např. kontrolovat duplicita
- CTX indexy – možno kombinovat „klasické“ kontextové indexy s XPATH výrazy, může vést k rychlejšímu vyhledávání, ale je třeba to chvíli ladit (spočítat statistiky tabulí, indexů, ...)

XMTYPE a XSLT

- Zaregistrujeme XML schéma
- Vytvoříme tabulku s XMLTYPEm nad tímto schématem
- Vytvoříme dokument s transformací (což je XMLTYPE) a vložíme do nějaké jiné tabule, která má XMLTYPE sloupec

```
SELECT
  XMLtransform(x.xmlcol,
               DBURIType('/XDB/STYLESHEET_TAB/ROW
                           [ID=1]/STYLESHEET/text()').getXML()).getStringVal()
  AS result
  FROM po_tab x;
```

Zdroje informací

ORACLE Documentation

http://www.oracle.com/technology/documentation/index.html