

A Hitchhikers Guide Integrating Oracle XML DB with 11gR2 and SQL Developer 3.2.2

Coleman Leviter, OCP
Arrow Electronics
IT Software Systems Engineer
cleviter@ieee.org

Good Housekeeping

Cell phones – AUDIBLE

Conversation – INSIDE

Enjoy the presentation





Wanted

Speakers

CV

- WMS Group – Fifteen years
- VAX Rewrite (.for) to UNIX (.c, .pc)
- VAX Forms to Oracle Forms
- TIFF file migration to Oracle
- XML Development
- WMS Development and Support
- IOUG Select Contributor
- ODTUG Journal
- IOUG Quick Tips
- NYOUG WEB SIG Chair, Steering Committee
- Oracle Open World
- IOUG Conference Committee
- IOUG Board of Directors

XML Xpath Terminology

- XPath is a language for finding information in an XML document. XPath is used to navigate through elements and attributes in an XML document. The primary purpose of XPath is to address parts of an XML document.
- Shred or extract parts of an XML Document
- The XML representation of schema components uses a vocabulary identified by the namespace name <http://www.w3.org/2001/XMLSchema>. For brevity, the text and examples in this specification use the prefix `xs:` to stand for this namespace; in practice, any prefix can be used.

`//author` - All `<author>` elements in the document.

`author/*` - All elements that are the children of `<author>` elements.

`author[1]` The first `<author>` element in the current context node.

`author[first-name][3]` - The third `<author>` element that has a `<first-name>` child.

`my:book` - The `<book>` element from the “my” namespace.

XML Xpath (shredding) Terminology

- `EXTRACT(XMLTYPE_instance, XPath_string)` or
`EXTRACT(XMLTYPE_instance, XPath_string, namespace_string)`

`EXTRACT (XML)` - It applies a `VARCHAR2` XPath string and returns an `XMLType` instance containing an XML fragment. You can specify an absolute XPath_string with an initial slash or a relative XPath_string by omitting the initial slash. If you omit the initial slash, the context of the relative path defaults to the root node. The optional namespace_string must resolve to a `VARCHAR2` value that specifies a default mapping or namespace mapping for prefixes, which Oracle Database uses when evaluating the XPath expression(s).

- `getClobVal()` - Returns a `CLOB` containing an XML document based on the contents of the `XMLType`.
- `XMLTYPE (CLOB)` – constructor; convert properly formed `CLOB` to `XMLTYPE`, raises an exception

Oracle9i has a dedicated XML datatype called `XMLTYPE`. It is made up of a `CLOB` to store the original XML data and a number of member functions to make the data available to SQL.

XPATH Terminology

- VALUE takes as its argument a correlation variable (table alias) associated with a row of an object table and returns object instances stored in the object table. The type of the object instances is the same type as the object table.
- Table functions are functions that produce a collection of rows (either a nested table or a varray) that can be queried like a physical database table or assigned to a PL/SQL collection variable. You can use a table function like the name of a database table, in the FROM clause of a query, or like a column name in the SELECT list of a query.

XPATH Terminology

- XMLSEQUENCE - XMLSEQUENCE(XMLTYPE_instance)
- XMLSEQUENCE operator is used to split multi-value results from XMLTYPE queries into multiple rows.
- The first form takes as input an XMLType instance and returns a varray of the top-level nodes in the XMLType. This form is effectively superseded by the SQL/XML standard function XMLTable, which provides for more readable SQL code. Prior to Oracle Database 10g Release 2, XMLSequence was used with SQL function TABLE to do some of what can now be done better with the XMLTable function.
- Because XMLSequence returns a collection of XMLType, you can use this function in a TABLE clause to unnest the collection values into multiple rows, which can in turn be further processed in the SQL query.

Xpath Expression Syntax

XML Document (fully qualified)

```
<?xml version="1.0" encoding="UTF-8"?> <!-- XML Declaration-->
```

```
<root>
```

```
  <parent>
```

```
    <child>robot</child>
```

```
    <child>ball</child>
```

```
  </parent>
```

```
  <parent>
```

```
    <child>airplane</child>
```

```
    <child>ipod</child>
```

```
  </parent>
```

```
</root>
```

Expression

Refers to

parent/child[1]

The first <child> of each <parent>.

(parent/child)[1]

The first <child> from the entire set of children of <parent> elements.

parent[1]/child[2]

The second <child> of the first <parent>.

parent/*

All elements that are the children of <parent> elements.

Namespaces Definition

- Why Namespaces – <http://www.w3.org/TR/REC-xml-names/>
- “Such documents, containing multiple markup vocabularies, pose problems of recognition and collision. Software modules need to be able to recognize the elements and attributes which they are designed to process, even in the face of "collisions" occurring when markup intended for some other software package uses the same element name or attribute name.
- These considerations require that document constructs should have names constructed so as to avoid clashes between names from different markup vocabularies. This specification describes a mechanism, *XML namespaces*, which accomplishes this by assigning expanded names to elements and attributes. “



- So, in simple terms, it's a technique to distinguish two elements with the same names from each other.
- The technique allows one to import (or define) another fully qualified XML document into an existing one without causing an element name collision.

XML Namespace Terms

- XML Namespaces - The “xmlns” Attribute

When using prefixes in XML, a so-called namespace for the prefix must be defined.

The namespace is defined by the xmlns attribute in the start tag of an element.

The namespace declaration has the following syntax.

`xmlns:prefix="URI"`.

- A Uniform Resource Identifier (URI) is a string of characters which identifies an Internet Resource.

The most common URI is the Uniform Resource Locator (URL) which identifies an Internet domain address.

SQL XML Query (SQLX) Terminology

- Create an SQL XML query
- Defined by ISO/IEC 9075-14:2003 ¹
- XMLAGG() is an aggregate function. It takes a collection of XML fragments and returns an aggregated XML document.
- XMLELEMENT() takes an element name for *identifier*, an optional collection of attributes (XMLATTRIBUTES) for the element, and arguments that make up the content of the element
- XMLFOREST() converts each of its argument parameters to XML, and then returns an XML fragment that is the concatenation of these converted arguments.

¹ ISO/IEC 9075-14:2003 defines ways in which Database Language SQL can be used in conjunction with XML. It defines ways of importing and storing XML data in an SQL database, manipulating it within the database and publishing both XML and conventional SQL-data in XML form

SQLX Examples – Table Contents (1_1)

```
SQL> select * from myobject;
```

THINGS	QUANTITY	PARENT
BALL	2	1
KEY	3	1
TABLE	1	1
FRISBEE	4	2
BBQ	1	2
SWITCH	6	2

6 rows selected.



SQLX - XMLELEMENT Example (1_2)

- XMLELEMENT() takes an element name for identifier, an optional collection of attributes for the element, and arguments that make up the content of the element

```
1 SELECT XMLELEMENT("thing", XMLATTRIBUTES( obj.quantity AS
   "QTY"), obj.things )
2 AS "Object_list"
3* FROM myobject obj
SQL> /
```

Object_list

```
<thing QTY="2">BALL</thing>
<thing QTY="3">KEY</thing>
<thing QTY="1">TABLE</thing>
<thing QTY="4">FRISBEE</thing>
<thing QTY="1">BBQ</thing>
<thing QTY="6">SWITCH</thing>
```

6 rows selected.

SQLX - XMLFOREST Example (1_3)

- XMLFOREST() converts each of its argument parameters to XML, and then returns an XML fragment that is the concatenation of these converted arguments.

```
SQL> SELECT XMLFOREST(obj.quantity, obj.things) "Object_list"  
2 FROM myobject obj;
```

Object_list

```
-----  
<QUANTITY>2</QUANTITY><THINGS>BALL</THINGS>  
<QUANTITY>3</QUANTITY><THINGS>KEY</THINGS>  
<QUANTITY>1</QUANTITY><THINGS>TABLE</THINGS>  
<QUANTITY>4</QUANTITY><THINGS>FRISBEE</THINGS>  
<QUANTITY>1</QUANTITY><THINGS>BBQ</THINGS>  
<QUANTITY>6</QUANTITY><THINGS>SWITCH</THINGS>
```

6 rows selected.

Prettyprint or Indenting

- Formatting Convention for Output
- XML Parent Child Relationship
- Changes from 10g, 11gR1, 11gR2
- `SELECT XMLSERIALIZE (DOCUMENT -
-line one
(<XML_Document >
• AS CLOB indent) [<alias >] from dual; --
line three`

SQLX - XMLAGG Example (1_4_3) (Pretty Print)

- XMLAGG() is an aggregate function. It takes a collection of XML fragments and returns an aggregated XML document.

```
1 SELECT XMLELEMENT("OBJECT",
   XMLAGG(XMLELEMENT("Things",
2 obj.things || ' ' ||obj.quantity ) ORDER BY obj.things)) AS "Object_list"
3 FROM myobject obj
SQL> /
```

Object_list

```
<OBJECT><Things>BALL 2</Things><Things>BBQ 1</Things>
<Things>FRISBEE 4</Things><Things>KEY
 3</Things><Things>SWITCH 6</Things><Things>TABLE
1</Things></OBJECT>
```

XML Document – TOAD OUT

The screenshot shows the TOAD interface with a text editor window titled "Text Editor: MSG_DATA_XML". The editor displays the following XML code:

```
<ns1:object_group xmlns:ns1="http://www.w3.org/2001/XMLSchema/s
  <obj1:object>
    <obj1:thing>ball</obj1:thing>
    <obj1:thing>key</obj1:thing>
    <obj1:thing>table</obj1:thing>
  </obj1:object>
  <obj2:object xmlns:obj2="http://www.w3.org/2001/XMLSchema/obj
    <obj2:thing>frisbee</obj2:thing>
    <obj2:thing>bbq</obj2:thing>
    <obj2:thing>switch</obj2:thing>
  </obj2:object>
</ns1:object_group>
```

The XML Tree view on the left shows the structure of the document:

- ns1:object_group
 - obj1:object
 - obj1:thing ball
 - obj1:thing key

The SubNodes of obj1:object table is as follows:

Name	Atts	Nodes	Value
obj1:thing	0	0	ball
obj1:thing	0	0	key
obj1:thing	0	0	table

The Attributes of obj1:object table is empty.

XML – SQL Developer OUT

The screenshot displays the Oracle SQL Developer interface. The main window shows a table named 'INBOUND_INTERFACE' with columns: TRAN_CODE, MSG_DATA, MSG_DATA_XML, MSG_DATA_CLOB, HOST_DATA_TYPE, DEL_MSG_DATA_CLOB, and DEL_MSG_DAT. The data rows show various test entries with XML content in the MSG_DATA and MSG_DATA_CLOB columns.

An 'XML Data' window is open, displaying the XML content from the selected row. The XML is as follows:

```
<?xml version="1.0" encoding="UTF-8"?>
<ns1:object_group xmlns:ns1="http://www.w3.org/2001/XMLSchema/sample_namespace_1" xmlns:obj1="http://www.w3.org/2001/XMLSchema/obj_1">
  <obj1:object>
    <obj1:thing>ball</obj1:thing>
    <obj1:thing>key</obj1:thing>
    <obj1:thing>table</obj1:thing>
  </obj1:object>
  <obj2:object xmlns:obj2="http://www.w3.org/2001/XMLSchema/obj_2">
    <obj2:thing>frisbee</obj2:thing>
    <obj2:thing>bbq</obj2:thing>
    <obj2:thing>switch</obj2:thing>
  </obj2:object>
</ns1:object_group>
```

The interface also shows a 'Connections' pane on the left with a tree view of database objects, and a 'SQL History' pane at the bottom. The status bar indicates 'Fetching Rows: 55' and 'Editing'.

SQLX Example – XMLAGG, XMLATTRIBUTES, Namespace (1_5)

DECLARE

```
lcl_obj1 CLOB; lcl_obj2 CLOB; lcl_full_xml CLOB;
```

BEGIN

```
SELECT XMLTYPE.getclobval(XMLELEMENT("obj1:object",
    xmlagg(xmlelement("obj1:thing",obj.things) )) )
```

```
INTO lcl_obj1 FROM myobject obj WHERE obj.parent = '1';
```

```
SELECT XMLTYPE.getclobval(XMLELEMENT("obj2:object",
    XMLATTRIBUTES ('http://www.w3.org/2001/XMLSchema/obj_2' AS "xmlns:obj2"), /* obj2 */
    XMLAGG(xmlelement("obj2:thing",obj.things) )) )
```

```
INTO lcl_obj2 FROM myobject obj WHERE obj.parent = '2';
```

```
SELECT ( '<?xml version="1.0" encoding="UTF-8"?>' ||
```

```
    '<ns1:object_group
xmlns:ns1="http://www.w3.org/2001/XMLSchema/sample_namespace_1"
xmlns:obj1="http://www.w3.org/2001/XMLSchema/obj_1">' ||          /* obj1 */
```

```
    lcl_obj1|| lcl_obj2|| '</ns1:object_group>' ) INTO lcl_full_xml FROM dual;
```

```
dbms_output.put_line(lcl_full_xml);
```

XML Namespace Example – table contents or in-line (1_6_1)

Table: my_xml_table; Column: xmlcol XMLTYPE
 Column: ref_id NUMBER

```
<?xml version="1.0" encoding="UTF-8"?>  
<ns1:object_group xmlns:ns1="http://www.w3.org/2001/XMLSchema/sample_namespace_1"  
  xmlns:obj1="http://www.w3.org/2001/XMLSchema/obj_1">  
  <obj1:object>  
    <obj1:thing>ball</obj1:thing>  
    <obj1:thing>key</obj1:thing>  
    <obj1:thing>table</obj1:thing>  
  </obj1:object>  
  <obj2:object xmlns:obj2="http://www.w3.org/2001/XMLSchema/obj_2">  
    <obj2:thing>frisbee</obj2:thing>  
    <obj2:thing>bbq</obj2:thing>  
    <obj2:thing>switch</obj2:thing>  
  </obj2:object>  
</ns1:object_group>
```

XML XPath – Cursor - obtain each row (1_7)

```
DECLARE
-- Cursor for parsing objects_group XML
CURSOR obj_cur
IS
  SELECT EXTRACT (VALUE
    (entire_things), '/*', 'xmlns:obj1="http://www.w3.org/2001/XMLSchema/obj_1"').getstringval() AS lcl_thing
  FROM my_xml_table mxt,
  TABLE(xmlsequence(extract(mxt.xmlcol,
    '//obj1:thing', 'xmlns:obj1="http://www.w3.org/2001/XMLSchema/obj_1"'))) entire_things
  WHERE mxt.ref_id = 1;
BEGIN
  FOR obj_row IN obj_cur
  LOOP
    dbms_output.put_line('each element thing ' || obj_row.lcl_thing );
  END LOOP;
END anonymous_block ;
```

```
each element thing <obj1:thing xmlns:obj1="http://www.w3.org/2001/XMLSchema/obj_1">ball</obj1:thing>
each element thing <obj1:thing xmlns:obj1="http://www.w3.org/2001/XMLSchema/obj_1">key</obj1:thing>
each element thing <obj1:thing xmlns:obj1="http://www.w3.org/2001/XMLSchema/obj_1">table</obj1:thing>
```

PL/SQL procedure successfully completed.

XPath – all data merged (1_8)

```
DECLARE
    -- Cursor for parsing objects_group XML
    CURSOR obj_cur
    IS
        SELECT EXTRACT (VALUE (entire_things),'/*/text()').getstringval () AS
lcl_thing
        FROM my_xml_table mxt,
            table(xmlsequence(extract(mxt.xmlcol, '*'))) entire_things
        WHERE mxt.ref_id = 1;
BEGIN
    FOR obj_row IN obj_cur LOOP
        dbms_output.put_line('each element thing || obj_row.lcl_thing ');
    END LOOP;
END anonymous_block ;
SQL> /
each element thing _ballkeytablefrisbeebbqswitch
PL/SQL procedure successfully completed.
```

XML Shredding (XPATH) Example (1_9)

- XML Document

```
<objects>
  <thing>ball</thing>
  <thing>key</thing>
  <thing>table</thing>
</objects>
```

-
- SQLX (XML Query Xpath)

```
Select value(tab).extract('/*').getStringVal() "This Column"
from table ( XMLSequence(extract (
  XMLType('<objects><thing>ball</thing><thing>key</thing><thing>table</thing></objects>'),'objects/
  *) ) ) tab;
```

-
- Results

This Column

```
<thing>ball</thing>
<thing>key</thing>
<thing>table</thing>
```


XPath Shredding Deux (1_10)

- XML Document

```
<objects>
  <thing>ball</thing>
  <thing>key</thing> ← extract this data item
  <thing>table</thing>
</objects>
```

=====

- SQLX (XML Query)

```
select value(tab).extract('/objects/thing[2]/text()').getStringVal() "This Column"
from table ( XMLSequence(extract (
XMLType('<objects><thing>ball</thing><thing>key</thing><thing>table</thing></objects>'),'*')
) tab;
```

=====

- Results

This Column

key

XML Xpath Shredding Namespace Failure 1 (1_11)

```
SQL> select
  value(tab).extract('//obj2:thing/text()','xmlns:obj1="http://www.w3.org/2001/XMLSchema
a/obj_1").getStringVal() "This_Column",
  value(tab).extract('//obj1:thing/text()','xmlns:obj1="http://www.w3.org/2001/XMLSchema
a/obj_1").getStringVal() "That_Column"
3 from table ( XMLSequence(extract
4 (XMLType('<?xml version="1.0" encoding="UTF-8"?><ns1:object_group
xmlns:ns1="http://www.w3.org/2001/XMLSchema/sample_namespace_1"
xmlns:obj1="http://www.w3.org/2001/XMLSchema/obj_1">
5 <obj1:object><obj1:thing>ball</obj1:thing><obj1:thing>key</obj1:thing>
6 <obj1:thing>table</obj1:thing></obj1:object>
7 <obj2:object xmlns:obj2="http://www.w3.org/2001/XMLSchema/obj_2">
8 <obj2:thing>frisbee</obj2:thing>
9 <obj2:thing>bbq</obj2:thing>
10 <obj2:thing>switch</obj2:thing>
11 </obj2:object>
12 </ns1:object_group>'),'*') ) ) tab;
```

XML Xpath Shredding Namespace Failure 2 (1_11)

```
select value(tab).extract('//obj2:thing/text()'
```

```
  'xmlns:obj1="http://www.w3.org/2001/XMLSchema/obj_1").getStringVal()  
  "This_Column",
```

*

ERROR at line 1:

ORA-31011: XML parsing failed

ORA-19202: Error occurred in XML processing

LPX-00601: Invalid token in: '//obj2:thing/text()' ← ob2 <>
ob1

ORA-06512: at "SYS.XMLTYPE", line 119

XML XPath Shredding 1 – data combined (1_12)

```
1 select
  value(tab).extract('//obj1:thing/text()','xmlns:obj1="http://www.w3.org/2001/XMLSchema/obj_1"').getStringVal()
  "This_Column",
  value(tab).extract('//obj1:thing/text()','xmlns:obj1="http://www.w3.org/2001/XMLSchema/obj_1"').getStringVal()
  "That_Column"
3 from table ( XMLSequence(extract
4 (XMLType('<?xml version="1.0" encoding="UTF-8"?><ns1:object_group
  xmlns:ns1="http://www.w3.org/2001/XMLSchema/sample_namespace_1"
  xmlns:obj1="http://www.w3.org/2001/XMLSchema/obj_1">
5 <obj1:object><obj1:thing>ball</obj1:thing><obj1:thing>key</obj1:thing>
6 <obj1:thing>table</obj1:thing></obj1:object>
7 <obj2:object xmlns:obj2="http://www.w3.org/2001/XMLSchema/obj_2">
8 <obj2:thing>frisbee</obj2:thing>
9 <obj2:thing>bbq</obj2:thing>
10 <obj2:thing>switch</obj2:thing>
11 </obj2:object>
12* </ns1:object_group>'),'*') ) ) tab
```

XML XPath Shredding 2 – data combined (1_12)

```
SQL> col this_column format a20;  
SQL> col that_column format a20;  
SQL> /
```

This_Column	That_Column
-----	-----
ballkeytable	ballkeytable

XML Example 1 (1_13)

```
select value(tab).extract('//obj1:thing/text()',
    'xmlns:obj1="http://www.w3.org/2001/XMLSchema/obj_1"').getStringVal()
    "This_Column",
    value(tab).extract('//obj2:thing/text()',
    'xmlns:obj2="http://www.w3.org/2001/XMLSchema/obj_2"').getStringVal()
    "That_Column"
4  from  table ( XMLSequence(extract
(XMLType('<?xml version="1.0" encoding="UTF-8"?><ns1:object_group
xmlns:ns1="http://www.w3.org/2001/XMLSchema/sample_namespace_1"
xmlns:obj1="http://www.w3.org/2001/XMLSchema/obj_1">
<obj1:object><obj1:thing>ball</obj1:thing><obj1:thing>key</obj1:thing>
7  <obj1:thing>table</obj1:thing></obj1:object>
    <obj2:object xmlns:obj2="http://www.w3.org/2001/XMLSchema/obj_2">
9  <obj2:thing>frisbee</obj2:thing>
10 <obj2:thing>bbq</obj2:thing>
11 <obj2:thing>switch</obj2:thing>
12 </obj2:object>
13* </ns1:object_group>'), '*' ) ) tab
```

XML Example 2 (1_13)

SQL> /

This_Column	That_Column
-----	-----
ballkeytable	frisbee bbqswitch

XML Xpath 1 – all data together (1_14_2)

```
SQL> select
  value(tab).extract('/*/text()','xmlns:obj1="http://www.w3.org/2001/XMLSchema/obj_1").getStringVal() "This_Column",
  value(tab).extract('//obj2:thing/text()','xmlns:obj2="http://www.w3.org/2001/XMLSchema/obj_2").getStringVal() "That_Column"
3 from table ( XMLSequence(extract
4 (XMLType('<?xml version="1.0" encoding="UTF-8"?>
5 <ns1:object_group xmlns:ns1="http://www.w3.org/2001/XMLSchema/sample_namespace_1"
  xmlns:obj1="http://www.w3.org/2001/XMLSchema/obj_1">
  <obj1:object><obj1:thing>ball</obj1:thing><obj1:thing>key</obj1:thing>
7 <obj1:thing>table</obj1:thing></obj1:object>
8 <obj2:object xmlns:obj2="http://www.w3.org/2001/XMLSchema/obj_2">
9 <obj2:thing>frisbee</obj2:thing>
10 <obj2:thing>bbq</obj2:thing>
11 <obj2:thing>switch</obj2:thing>
12 </obj2:object>
13 </ns1:object_group>'),/'*') ) ) tab;
```


XML Xpath 2 – all data together (1_14_2)

This_Column

That_Column

ballkeytablefrisbee bbqswitch frisbee bbqswitch

Xpath 1 – extract data (1_15)

SQL> select

```
value(tab).extract('//obj1:thing[2]/text()','xmlns:obj1="http://www.w3.org/2001/XMLSchema/obj_1"').getStringVal() "This_Column",  
value(tab).extract('//obj2:thing/text()','xmlns:obj2="http://www.w3.org/2001/XMLSchema/obj_2"').getStringVal() "That_Column"
```

```
3 from table ( XMLSequence(extract
```

```
4 (XMLType('<?xml version="1.0" encoding="UTF-8"?>
```

```
5 <ns1:object_group xmlns:ns1="http://www.w3.org/2001/XMLSchema/sample_namespace_1"  
  xmlns:obj1="http://www.w3.org/2001/XMLSchema/obj_1">
```

```
6 <obj1:object><obj1:thing>ball</obj1:thing><obj1:thing>key</obj1:thing>
```

```
7 <obj1:thing>table</obj1:thing></obj1:object>
```

```
8 <obj2:object xmlns:obj2="http://www.w3.org/2001/XMLSchema/obj_2">
```

```
9 <obj2:thing>frisbee</obj2:thing>
```

```
10 <obj2:thing>bbq</obj2:thing>
```

```
11 <obj2:thing>switch</obj2:thing>
```

```
12 </obj2:object>
```

```
13 </ns1:object_group>'),'/*') ) ) tab;
```

Xpath 2 – extract data (1_15)

```
SQL> col this_column format a40;
```

```
SQL> col that_column format a40;
```

This_Column

That_Column

key

frisbeebbqswitch

Xpath Parsing Error 1 (1_16_3)

```
SELECT extract(value(t), '//text()').getStringVal() AS  
obj_examp  
FROM my_xml_table mxt,  
TABLE(xmlsequence(extract(mxt.xmlcol, '*'))) t  
WHERE mxt.ref_id = 1
```

OBJ_EXAMP

ballkeytablefrisbeebbqswitch

Xpath Parsing Error 2 (1_16_3)

```
SELECT    extract(value(t), '//obj1:thing/text()').getStringVal() AS
obj_examp
FROM my_xml_table,
table(xmlsequence(extract(xmlcol, '*'))) t
WHERE ref_id = 1
```

```
SELECT extract(value(t), '//obj1:thing/text()').getStringVal() AS
obj_examp
```

*

ERROR at line 1:

ORA-31011: XML parsing failed

ORA-19202: Error occurred in XML processing

LPX-00601: Invalid token in: '//obj1:thing/text()' ← namespace
definition undefined

XML XPATH – no unique elements (1_17)

```
SELECT  extract(value(t),'//obj1:thing/text()',
    'xmlns:obj1="http://www.w3.org/2001/XMLSchema/obj_1"').getStringV
    al() AS obj_example
FROM my_xml_table,
    table(xmlsequence(extract(xmlcol, '*'))) t
WHERE ref_id = 1
```

OBJ_EXAMPLE

ballkeytable

XML XPATH 1 – parse each element, using table (1_18)

```
DECLARE
```

```
-- Cursor for parsing objects_group XML
```

```
CURSOR obj_cur
```

```
IS
```

```
SELECT EXTRACT (VALUE (entire_things), '//obj1:thing/text()',  
  'xmlns:obj1="http://www.w3.org/2001/XMLSchema/obj_1"').  
  getstringval() AS lcl_thing
```

```
FROM my_xml_table,
```

```
TABLE(xmlsequence(extract(xmlcol, '//obj1:thing', -- data  
  xmlns:obj1="http://www.w3.org/2001/XMLSchema/obj_1")))  
entire_things
```

```
WHERE ref_id = 1;
```

XML XPATH 2 – parse each element, using table (1_18)

```
BEGIN
```

```
  FOR obj_row IN obj_cur  LOOP
```

```
    dbms_output.put_line
```

```
      ('each element thing ' || obj_row.lcl_thing );
```

```
  END LOOP;
```

```
END anonymous_block ;
```

```
each element thing ball
```

```
each element thing key
```

```
each element thing table
```


UPDATEXML – 1st (1_19_2)

```
SQL> describe my_xml_table;
```

Name	Null?	Type
-----	-----	-----
REF_ID		NUMBER
XMLCOL		XMLTYPE

```
select xmlcol from my_xml_table where ref_id = 1;
```

```
XMLCOL
```

```
-----  
<?xml version="1.0" encoding="UTF-8"?>  
<ns1:object_group xmlns:ns1=http://www.w3.org/2001/XMLSchema/sample_namespace_1  
                  xmlns:obj1="http://www.w3.org/2001/XMLSchema/obj_1">  
  <obj1:object>  
    <obj1:thing>ball</obj1:thing>  
    <obj1:thing>key</obj1:thing>  
    <obj1:thing>table</obj1:thing>  
  </obj1:object>  
  <obj2:object xmlns:obj2="http://www.w3.org/2001/XMLSchema/obj_2">  
    <obj2:thing>frisbee</obj2:thing>  
    <obj2:thing>bbq</obj2:thing>  
    <obj2:thing>switch</obj2:thing>  
  </obj2:object>  
</ns1:object_group>
```

UPDATEXML 1 – 2nd (1_19_2)

```
SQL> UPDATE my_xml_table mxt
      SET mxt.xmlcol = UPDATEXML(mxt.xmlcol,
                                '//obj2:thing[1]/text()', 'FRISBEE',
                                'xmlns:obj2=
"http://www.w3.org/2001/XMLSchema/obj_2"')
      WHERE mxt.ref_id = 1;

1 row updated.
```

-- CHANGE TO CAPS

UPDATEXML 2 – 2nd (1_19_2)

```
select xmlcol from my_xml_table where ref_id = 1;
```

XMLCOL

```
<?xml version="1.0" encoding="UTF-8"?>
```

```
<ns1:object_group xmlns:ns1=http://www.w3.org/2001/XMLSchema/sample_namespace_1  
  xmlns:obj1="http://www.w3.org/2001/XMLSchema/obj_1">
```

```
  <obj1:object>
```

```
    <obj1:thing>ball</obj1:thing>
```

```
    <obj1:thing>key</obj1:thing>
```

```
    <obj1:thing>table</obj1:thing>
```

```
  </obj1:object>
```

```
  <obj2:object xmlns:obj2="http://www.w3.org/2001/XMLSchema/obj_2">
```

```
    <obj2:thing>FRISBEE</obj2:thing>
```

```
    <obj2:thing>bbq</obj2:thing>
```

```
    <obj2:thing>switch</obj2:thing>
```

```
  </obj2:object>
```

```
</ns1:object_group>
```

XML XPATH 1 – parse each element, w/o table (1_20)

-- Anonymous Block

DECLARE

-- Cursor for parsing object_group XML

CURSOR obj_cur

IS SELECT EXTRACT (VALUE (entire_things),

 '//obj1:thing/text()',

 'xmlns:obj1="http://www.w3.org/2001/XMLSchema/obj_1"').

 getstringval() AS lcl_thing

FROM TABLE (XMLSequence(

 extract (XMLType('<?xml version="1.0" encoding="UTF-8"?>

<ns1:object_group xmlns:ns1=

 http://www.w3.org/2001/XMLSchema/sample_namespace_1

 xmlns:obj1="http://www.w3.org/2001/XMLSchema/obj_1">

XML XPATH 2 – parse each element, w/o table (1_20)

```
<obj1:object>
```

```
  <obj1:thing>ball</obj1:thing>
```

```
  <obj1:thing>key</obj1:thing>
```

```
  <obj1:thing>table</obj1:thing>
```

```
</obj1:object>
```

```
<obj2:object xmlns:obj2="http://www.w3.org/2001/XMLSchema/obj_2">
```

```
  <obj2:thing>frisbee</obj2:thing>
```

```
  <obj2:thing>bbq</obj2:thing>
```

```
  <obj2:thing>switch</obj2:thing>
```

```
</obj2:object>
```

```
</ns1:object_group>'),
```

```
'//obj1:thing', 'xmlns:obj1="http://www.w3.org/2001/XMLSchema/obj_1"' ) ) )
```

```
entire_things;
```

XML XPATH 3 – parse each element, w/o table (1_20)

```
BEGIN
```

```
    FOR obj_row IN obj_cur LOOP
```

```
        dbms_output.put_line('each element thing '||  
                               obj_row.lcl_thing );
```

```
    END LOOP;
```

```
END anonymous_block;
```

```
SQL> /
```

```
each element thing  ball
```

```
each element thing  key
```

```
each element thing  table
```

```
PL/SQL procedure successfully completed.
```

Oracle Text - XML Data Queries (1_21)

- Definitions
- “contains” operator is used when index is of type context
- returns a relevance score for every row selected
- ```
select ref_id,score (1) , xmlcol
from my_xml_table
where contains (xmlcol, 'table',1) > 0
```
- Demo

# DBMS\_XML.CONVERT (1\_22\_3)

- DBMS\_XMLGEN.CONVERT - converts XML data into the escaped or unescapes  
XML equivalent
- Example: the escaped form of the character “>” (without the “ characters) is “&gt;”  
(without the “ characters).
- Oracle’s suggested “workaround” was to:
  - 1) append a space (CHR(32)) to the end of the XPath string, then
  - 2) DBMS\_XMLGEN.CONVERT the string, finally
  - 3) TRIM the string. The workaround resolved the issue.



# DBMS\_XMLDOM (1\_23)

- Parse using XPath – define expression

A whole heap of Metadata

```
<?xml version="1.0" encoding="WINDOWS-1252"?>
<StandardHeader>
 <DateTime>12/08/2010 16:03:46</DateTime>
 <PacketNumber>1</PacketNumber>
 <AnyMorePackets>Y</AnyMorePackets>
 <TotalPackets>4</TotalPackets>
 <house>
 <room>
 <room_name name="kitchen"/>
 <room_item item="sink"/>
 </room>
 |
 |
 <room>
 <room_name name="kitchen"/>
 <room_item item="table"/>
 </room>
 </house>
</StandardHeader>
```

# DBMS\_XMLDOM (1\_24) (cont'd)

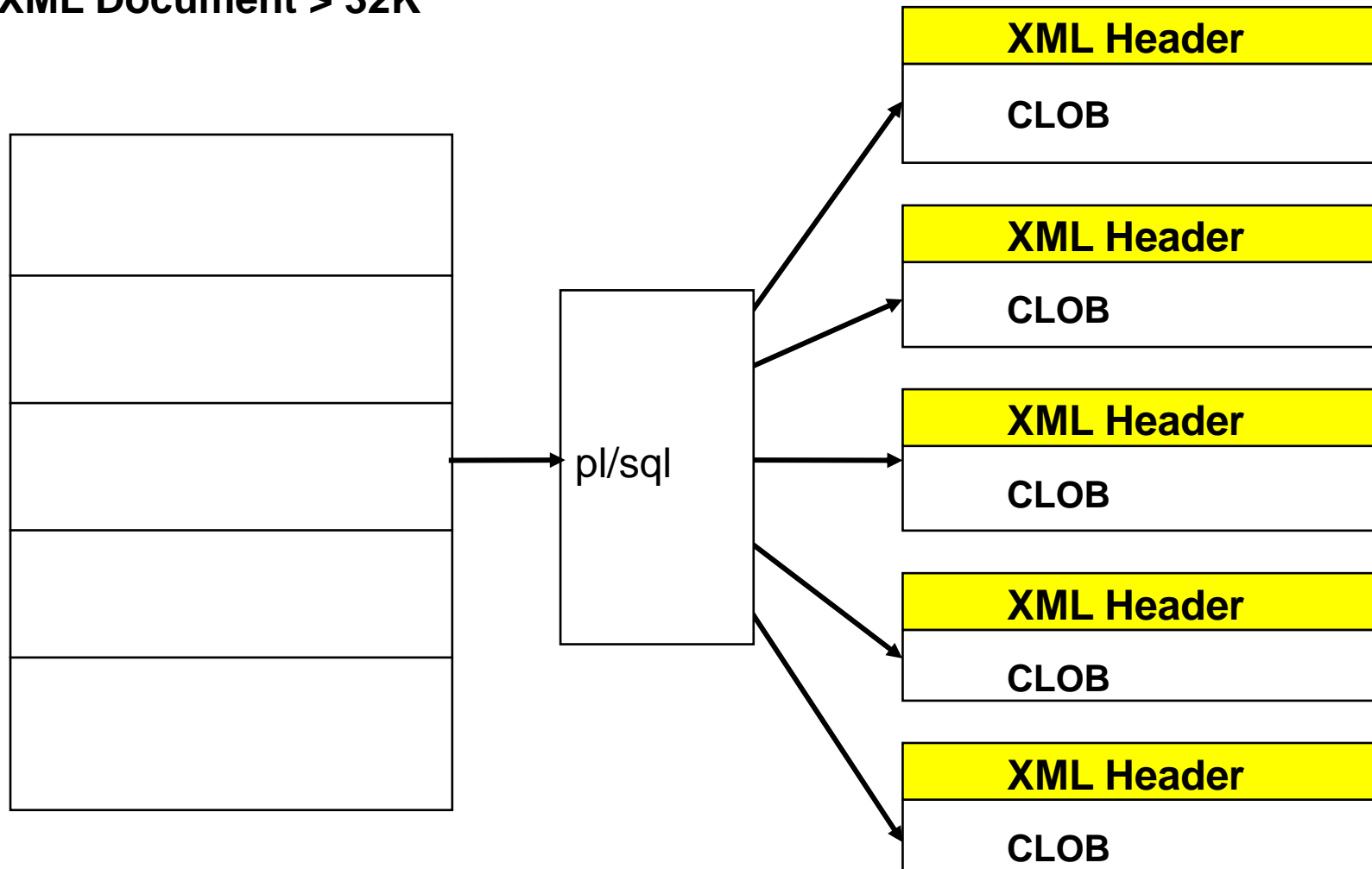
- Change XML and reduce metadata

```
<?xml version="1.0" encoding="WINDOWS-1252"?>
<StandardHeader>
 <DateTime>12/08/2010 16:03:46</DateTime>
 <PacketNumber>1</PacketNumber>
 <AnyMorePackets>Y</AnyMorePackets>
 <TotalPackets>4</TotalPackets>
<house>
 <room room_name="kitchen" room_item="sink"/>
 <room room_name="kitchen" room_item="table"/>
 <room room_name="kitchen" room_item="counter"/>
 <room room_name="kitchen" room_item="microwave"/>
 <room room_name="kitchen" room_item="oven"/>
 <room room_name="kitchen" room_item="range"/>
 <room room_name="kitchen" room_item="refrigerator"/>
 <room room_name="living_room" room_item="pictures"/>
 <room room_name="living_room" room_item="chair"/>
 <room room_name="living_room" room_item="pictures"/>
 <room room_name="living_room" room_item="HDTV"/>
 <room room_name="living_room" room_item="couch"/>
 <room room_name="living_room" room_item="chandelier"/>
 <room room_name="den" room_item="surround_sound"/>
 <room room_name="den" room_item="fireplace"/>
 <room room_name="den" room_item="table"/>
 <room room_name="den" room_item="chair"/>
 <room room_name="den" room_item="lamp"/>
 <room room_name="den" room_item="étagère"/>
 <room room_name="den" room_item="HDTV"/>
</house>
</StandardHeader>
```

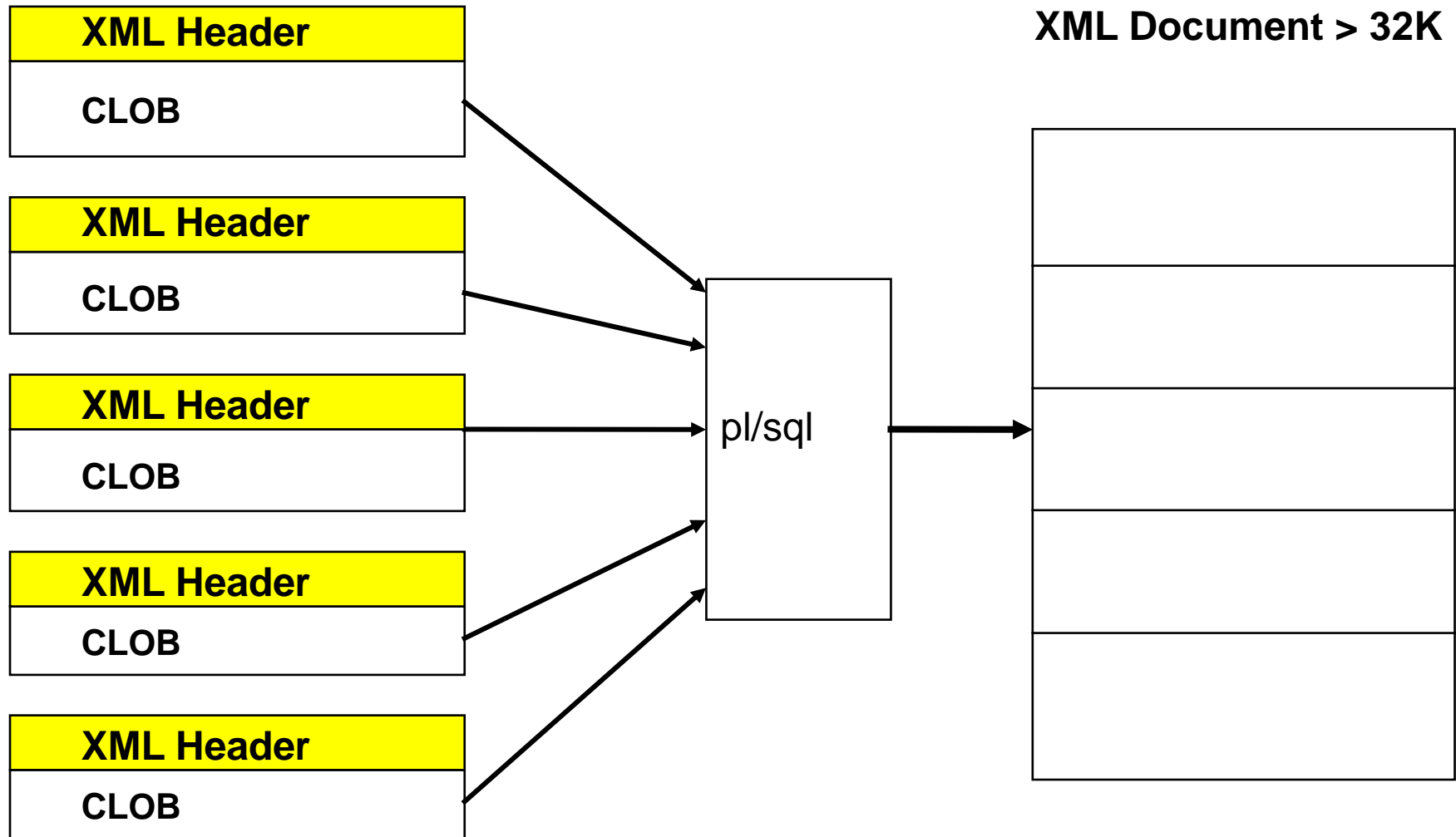
# XML Packet Creation

XML Document > 32K

CLOB < 32 K



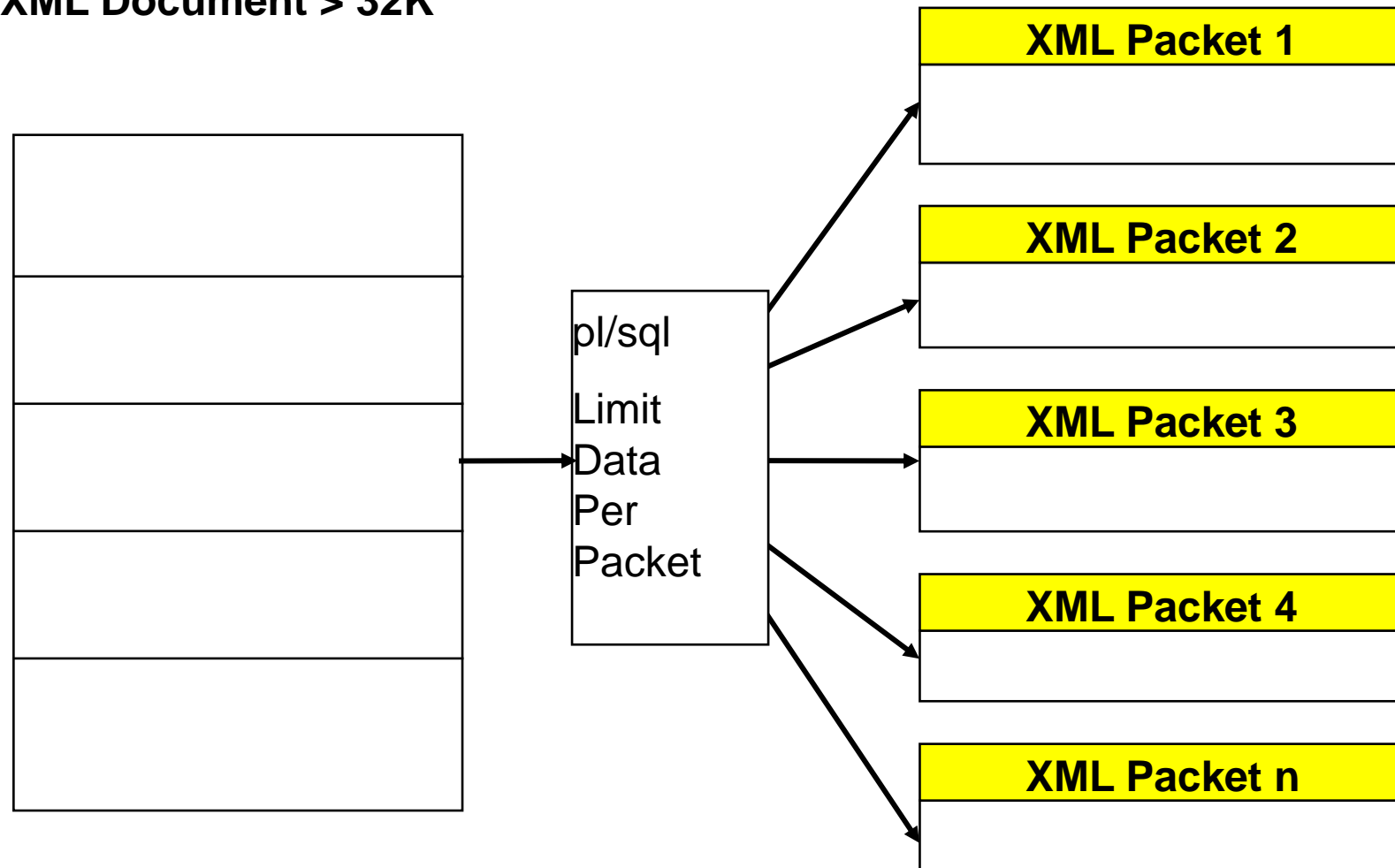
# XML Document Construction



# XML Packet Creation

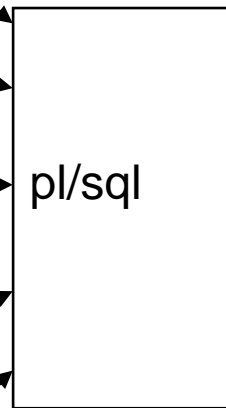
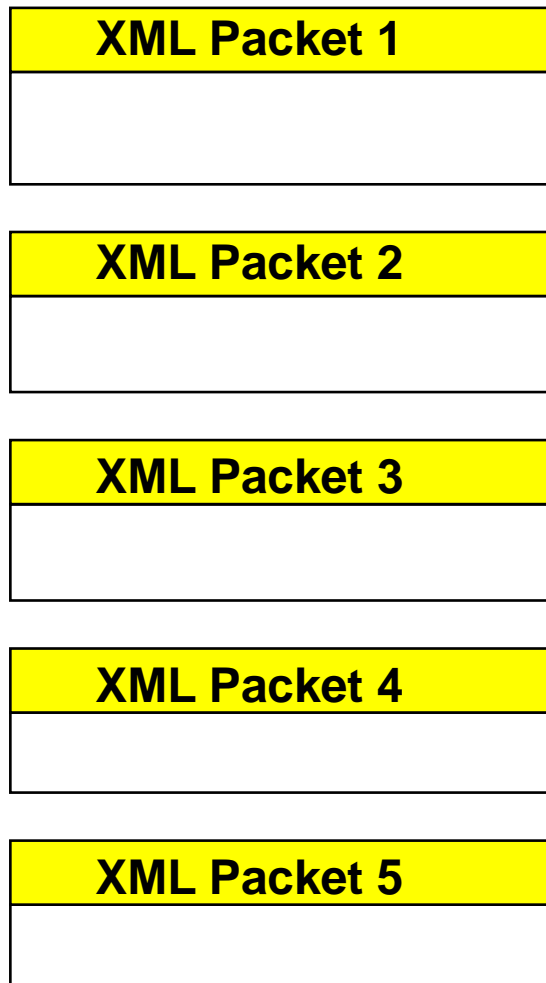
XML Document > 32K

XML Packet < 32 K

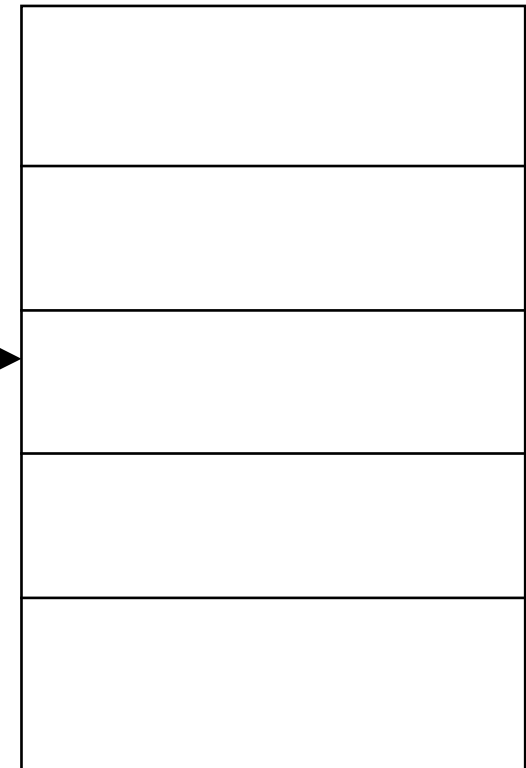


# XML Document Construction

XML Packet < 32 K



XML Document > 32K



# XQUERY (example)

```
SELECT xmlquery('xs:integer("12") 'returning content) aa from dual
```

```
SELECT xmlquery('
for $i in ora:view("my_relatabl") return $i
returning content) from dual
```

```
SELECT XMLQUERY(' (: run statement SQL Developer :)
<result>{
for $i in ora:view("my_relatabl") return $i
}</result>'
returning content) aa from dual
```

```
SELECT XMLQUERY(' (: run statement SQL Developer :)
<result>{
for $i in ora:view("my_relatabl")
where $i/ROW/NAME eq "kitchen"
return $i
}</result>'
returning content) aa from dual
```

# Bonus

- This presentation
- LibreOffice – The Document Foundation  
Vers 4.0.0.3  
Website: <http://www.libreoffice.org/#0>
- Sql Developer 3.2.2 – Cloud Connections



# References 1

Design Tools: XMLSPY – 30 day free trial

<http://www.altova.com/>

Oracle XML DB Docs: full implementation plus extensions of XML

<http://www.oracle.com/technetwork/database/features/xmlldb/index.html>

Oracle's SQL 11g Doc

<http://www.oracle.com/pls/db111/homepage>

Oracle Database 11g XML & SQL – Oracle Press

Wikipedia.com (xpath, xml, xmlelement, etc.)

Overview XML Constructs

[http://www.oradev.com/xml\\_functions.jsp](http://www.oradev.com/xml_functions.jsp)

## References 2

W3.org

W3schools.com

Oracle Database 10g XML & SQL: Design, Build, & Manage XML Applications in Java, C, C++, & PL/SQL (Osborne ORACLE Press Series) (Paperback) by Mark Scardina (Author), Ben Chang (Author), Jinyu Wang (Author)

<http://docs.oracle.com/javase/tutorial/reallybigindex.html>

iso.org

Sql Developer 3.2.2 - 12c Database Support

<http://www.oracle.com/technetwork/developer-tools/sql-developer/sqldev-newfeatures32-1731840.html>

# Please Note

Speaker: Coleman Leviter

Topic: A HitchhikersGuide Integrating Oracle XML DB with  
11gR2 and SQL Developer 3.2.2

## Contact:

[cleviter@ieee.org](mailto:cleviter@ieee.org)

Please enter “NYOUG” in the subject line

# Questions

