

Integrating Oracle 10g XML: A Case Study Part II

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

CV

- WMS Group – Ten 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 '06, '08
- NYOUG WEB SIG Chair
- IOUG Collaborate '07, '08, '09

ORACLE®
Certified Professional

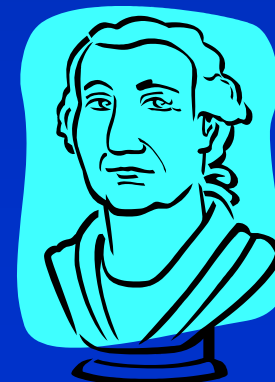
Presentation Objectives

- Oracle – XML
- Terminology
- Project Overview – EBS/SOA - WMS
- WMS Architecture Changes
- XSD Design
- XML Terminology
- XML Development
- XML XPATH, SQLX Examples
- Questions



Oracle XML History

- 8i 1998 - XML Api
- 9i 2001 - XML Storage
- 10g 2004 - XPath
- 11g 2007 - Binary XML
- 12g ?????



Terminology

- XML¹ - Short for eXtensible Markup Language, a specification developed by the World Wide Web Consortium (W3C). It allows designers to create their own customized tags, enabling the definition, transmission, validation, and interpretation of data between applications and between organizations.
- Warehouse Management System - a component of the movement and storage of materials within a warehouse
- Service Oriented Architecture² - (SOA) provides methods for systems development and integration where systems group functionality around business processes and package these as interoperable services.

¹ <http://en.wikipedia.org/wiki/XML>

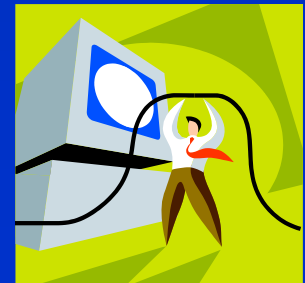
² http://en.wikipedia.org/wiki/Service-oriented_architecture



WMS Architecture Changes

- Maintain existing fixed length messages – production environment
- Detect incoming XML varying length messages – development
- Convert fixed length data messages to XML varying length data messages - migration
- Follow OAGIS 9.2 Specification - **O**pen **A**pplications **G**roup **I**ntegration **S**pecification
- OAGIS - is an effort to provide a canonical¹ business language for information integration. It uses XML as the common alphabet for defining business messages, and for identifying business processes (scenarios) that allow businesses and business applications to communicate. Not only is OAGIS the most complete set of XML business messages currently available, but it also accommodates the additional requirements of specific industries by partnering with various vertical industry groups.

¹ Wikipedia, conforming to orthodox or well-established rules or patterns, as of procedure

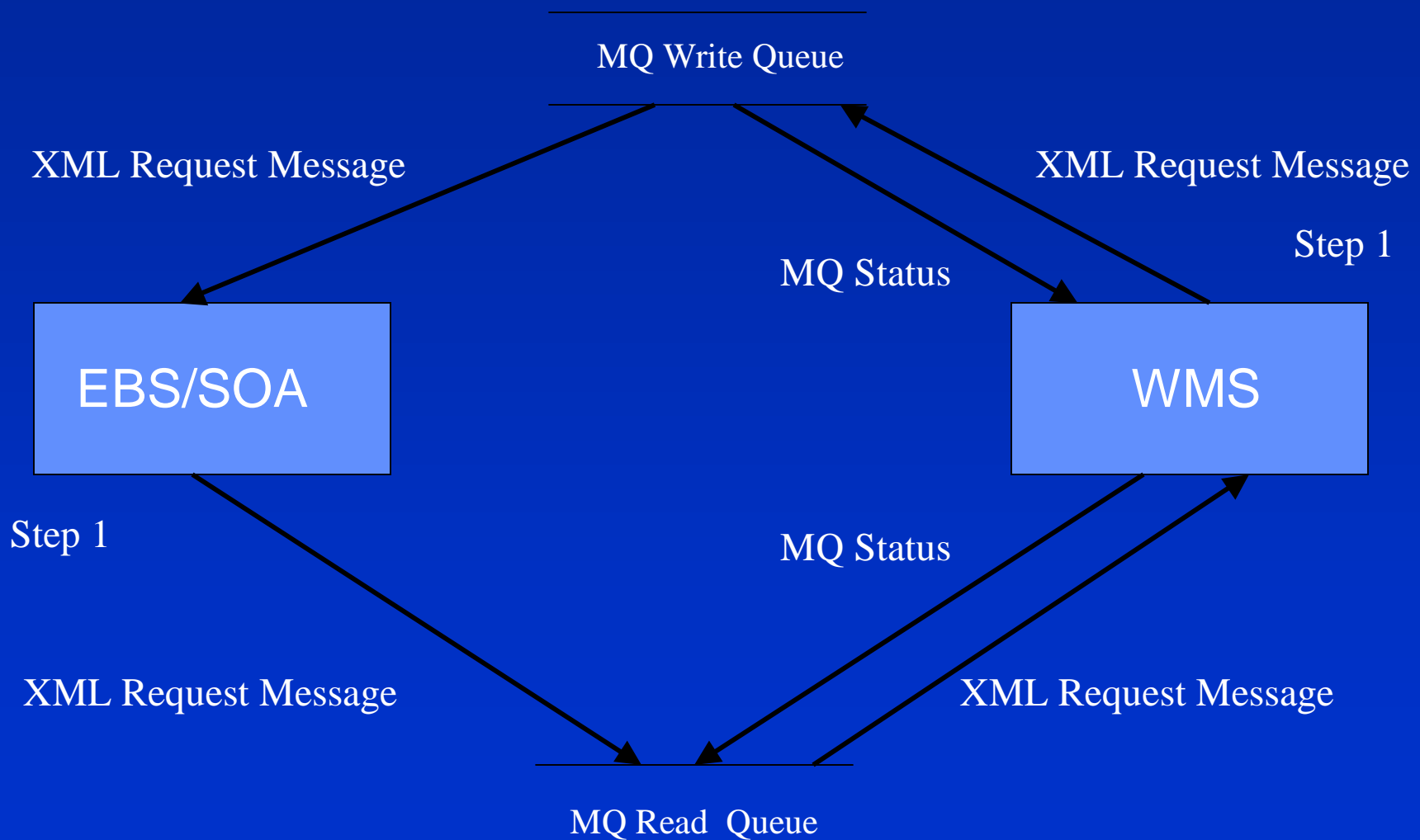


Project Steps

- Develop XSD
- Use Oracle 10g XML DB for messages (leverage 9i development)
- Use MQ Series for the communications layer – guaranteed message delivery
- Oracle Database 10g Enterprise Edition Release 10.2.0.1.0 - 64bi
PL/SQL Release 10.2.0.1.0 – Production

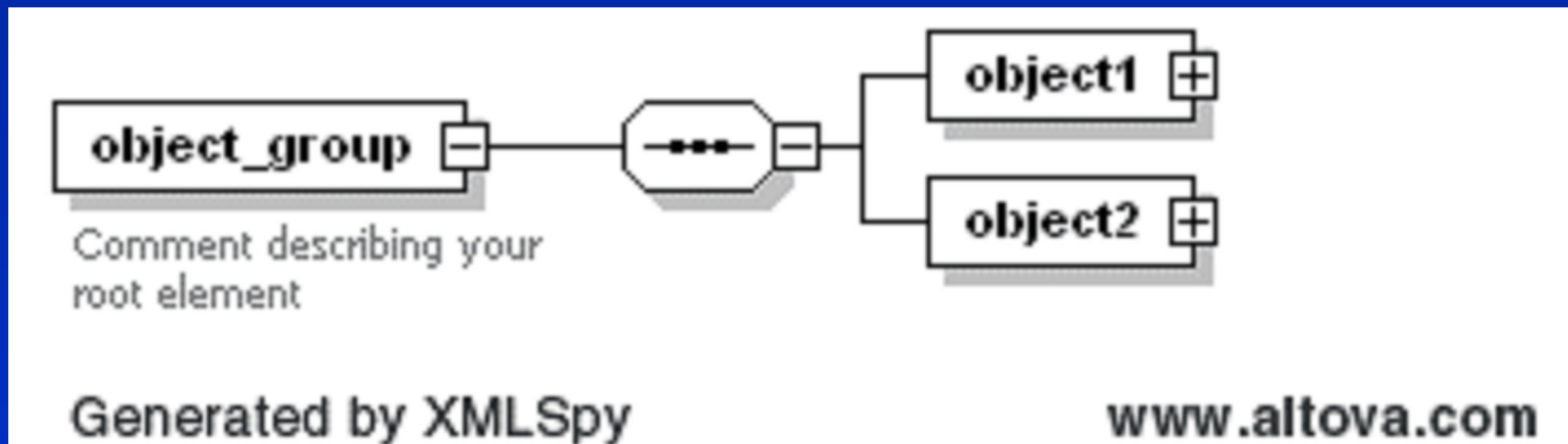


EBS/SOA-WMS Data Flow



XSD Design – XML SPY

- XML Schema Definition (XSD) language is the current standard schema language for all XML documents and data.



XSD Text View – XML SPY

```
<?xml version="1.0" encoding="UTF-8"?>
<object_group xmlns:xsi=
    "http://www.w3.org/2001/XMLSchema-instance"
    xsi:noNamespaceSchemaLocation=
    "G:\personal_files\ioug_collaborate_09\xml_example_object_thing.xsd">
  <object1>
    <thing/>
  </object1>
  <object2>
    <thing/>
  </object2>
</object_group>
```

Text Editor: MSG_DATA_XML

MSG_DATA_XML

XML Tree

Tag	Value
object_group	
object1	
thing	key
object2	
thing	desk

SubNodes

Name	Atts	Nodes	Value

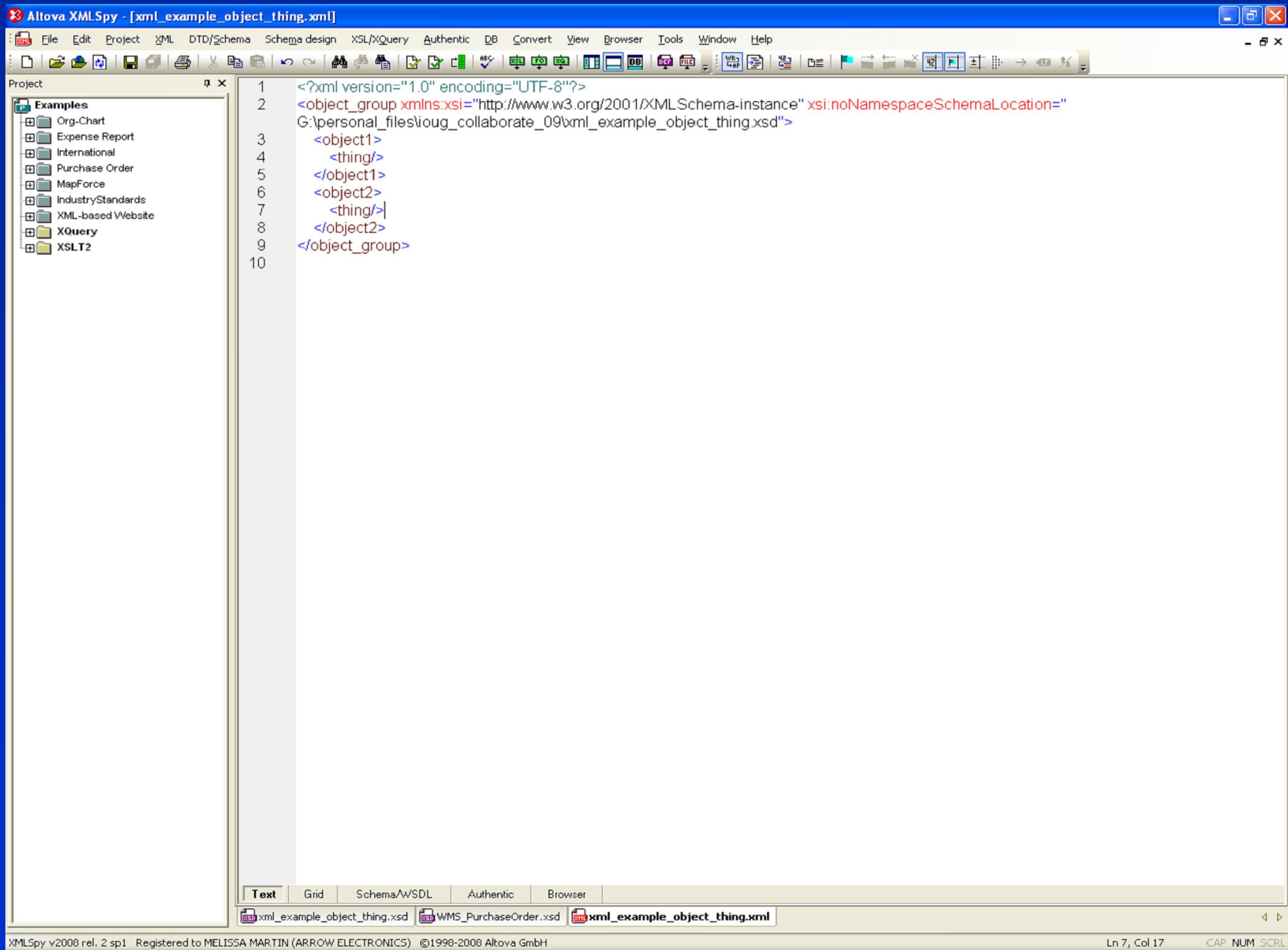
Attributes

Name	Value

```
<?xml version="1.0" encoding="UTF-8"?>
<object_group xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:noNamespaceSchemaLocation="G:\personal
<object1>
  <thing>key</thing>
</object1>
<object2>
  <thing>desk</thing>
</object2>
</object_group>
```

7: 24 Modified

AutoCommit is OFF | CAPS NUM INS



XPath Definition - Shredding

- **XPath** (XML Path Language) is an expression language for extracting portions of an XML document, or for calculating values (strings, numbers, or Boolean values) based on the content of an XML document. Used for document shredding. Insert data into relational table.

- **Expression** **Description**

<i>nodename</i>	Selects all child nodes of the named node
/	Selects from the root node
//	Selects all nodes in the document from the current node that match the selection no matter where they are
.	Selects the current node
..	Selects the parent of the current node
@	Selects attributes

- /A/B[1] Select the first B node from the A node

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.
- 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.
- 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.

`//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

parent/child[1]
(parent/child)[1]
parent[1]/child[2]
parent/*

Refers to

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

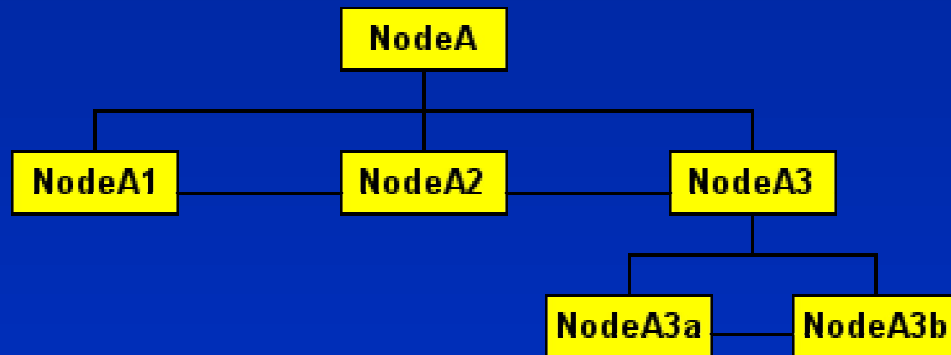
- Create an SQL XML query
 - Defined by ISO/IEC 9075-14:2003 ¹
 - XMLAGG() is an aggregate function. It takes a collection of XML fragments and returns an aggregated XML document.
 - XMLELEMENT() takes an element name for *identifier*, an optional collection of attributes (XMLATTRIBUTES) for the element, 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

Document Object Model (DOM)

- XML access defined as a tree structure
- Available with Oracle's 10g XDK (XML Developer's Kit)
JAVA/C/C+

- Navigation

```
NodeA.firstChild = NodeA1
NodeA.lastChild = NodeA3
NodeA.childNodes.length = 3
NodeA.childNodes[0] = NodeA1
NodeA.childNodes[1] = NodeA2
NodeA.childNodes[2] = NodeA3
NodeA1.parentNode = NodeA
NodeA1.nextSibling = NodeA2
NodeA3.prevSibling = NodeA2
NodeA3.nextSibling = null
NodeA.lastChild.firstChild = NodeA3a
NodeA3b.parentNode.parentNode = NodeA
```



- Methods

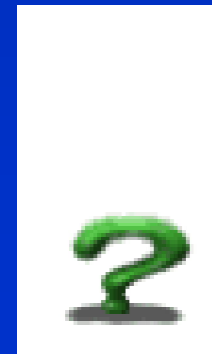
```
insertBefore()
replaceChild()
removeChild()
appendChild()
cloneNode()
```

SQLX Examples – Table Contents

```
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

- 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

- 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.
```

SQLX - XMLAGG Example

- 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

The screenshot shows the TOAD application window titled "TOAD - [WMS_DEV@RENT Schema Browser (WMS_DEV)]". A "Text Editor: MSG_DATA_XML" window is open, displaying an XML document. The XML content is as follows:

```
<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)
 - obj1:thing (table)
 - obj2:object
 - obj2:thing (frisbee)
 - obj2:thing (bbq)
 - obj2:thing (switch)

The "SubNodes of obj1:object" table is shown below:

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

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, showing the XML content of 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

```
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">' ||  
    lcl_obj1 || lcl_obj2 || '</ns1:object_group>' ) INTO lcl_full_xml FROM dual;  
dbms_output.put_line(lcl_full_xml);
```

```
END;
```

XML Namespace Example – table contents or in-line

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

```
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

```
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.
```

```
SQL>
```


XML Shredding (XPATH) Example

- 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

- 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

```
SQL> select
  value(tab).extract('//obj2:thing/text()','xmlns:obj1="http://www.w3.org/2001/XMLSchema/obj_1"').getStringVal()
  "This_Column",
2     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;
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 – data combined

```
1 select value(tab).extract('//obj1:thing/text()','xmlns:obj1="http://www.w3.org/2001/XMLSchema/obj_1"').getStringVal()
   "This_Column",
2     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
SQL> col this_column format a20;
SQL> col that_column format a20;
SQL> /
```

This_Column	That_Column
ballkeytable	ballkeytable

XML Example

```
1  select
  value(tab).extract('//obj1:thing/text()','xmlns:obj1="http://www.w3.org/2001/XMLSchema/obj_1"').getStringVal() "This_Column",
2
  value(tab).extract('//obj2:thing/text()','xmlns:obj2="http://www.w3.org/2001/XMLSchema/obj_2"').getStringVal()
3  "That_Column"
4  from  table ( XMLSequence(extract
5  (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">
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
```

SQL> /

This_Column	That_Column
ballkeytable	frisbeebbqswitch

XML Xpath – all data together

```
SQL> col this_column format a40;
SQL> col that_column format a40;
SQL> select value(tab).extract('/*/text()','xmlns:obj1="http://www.w3.org/2001/XMLSchema/obj_1"').getStringVal()
   "This_Column",
   2 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;
```

This_Column

That_Column

ballkeytablefrisbeebbqswitch frisbeebbqswitch

Xpath – extract data

```
SQL> col this_column format a40;
SQL> col that_column format a40;
SQL> select
  value(tab).extract('//obj1:thing[2]/text()','xmlns:obj1="http://www.w3.org/2001/XMLSchema/obj_1"').getStringVal() "This_Column",
  2     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"
  6   xmlns:obj1="http://www.w3.org/2001/XMLSchema/
obj_1">
  7   <obj1:object><obj1:thing>ball</obj1:thing><obj1:thing>key</obj1:thing>
  8   <obj1:thing>table</obj1:thing></obj1:object>
  9   <obj2:object xmlns:obj2="http://www.w3.org/2001/XMLSchema/obj_2">
 10   <obj2:thing>frisbee</obj2:thing>
 11   <obj2:thing>bbq</obj2:thing>
 12   <obj2:thing>switch</obj2:thing>
 13   </obj2:object>
 14 </ns1:object_group>'),'/*') ) ) tab;
```

This_Column

That_Column

key

frisbeebbqswitch

Xpath Parsing Error

```
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

ballkeytablefrisbeebqswitch

SQL>

```
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

SQL>

XML XPATH – no unique elements

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

OBJ_EXAMPLE

ballkeytable

XML XPATH – parse each element, using table

```
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', 'xmlns:obj1="http://www.w3.org/2001/XMLSchema/obj_1"')))
  entire_things
  WHERE 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  ball
each element thing  key
each element thing  table
```

PL/SQL procedure successfully completed.

UPDATEXML – 1st

```
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 – 2nd

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

1 row updated.

```
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>
=====
```

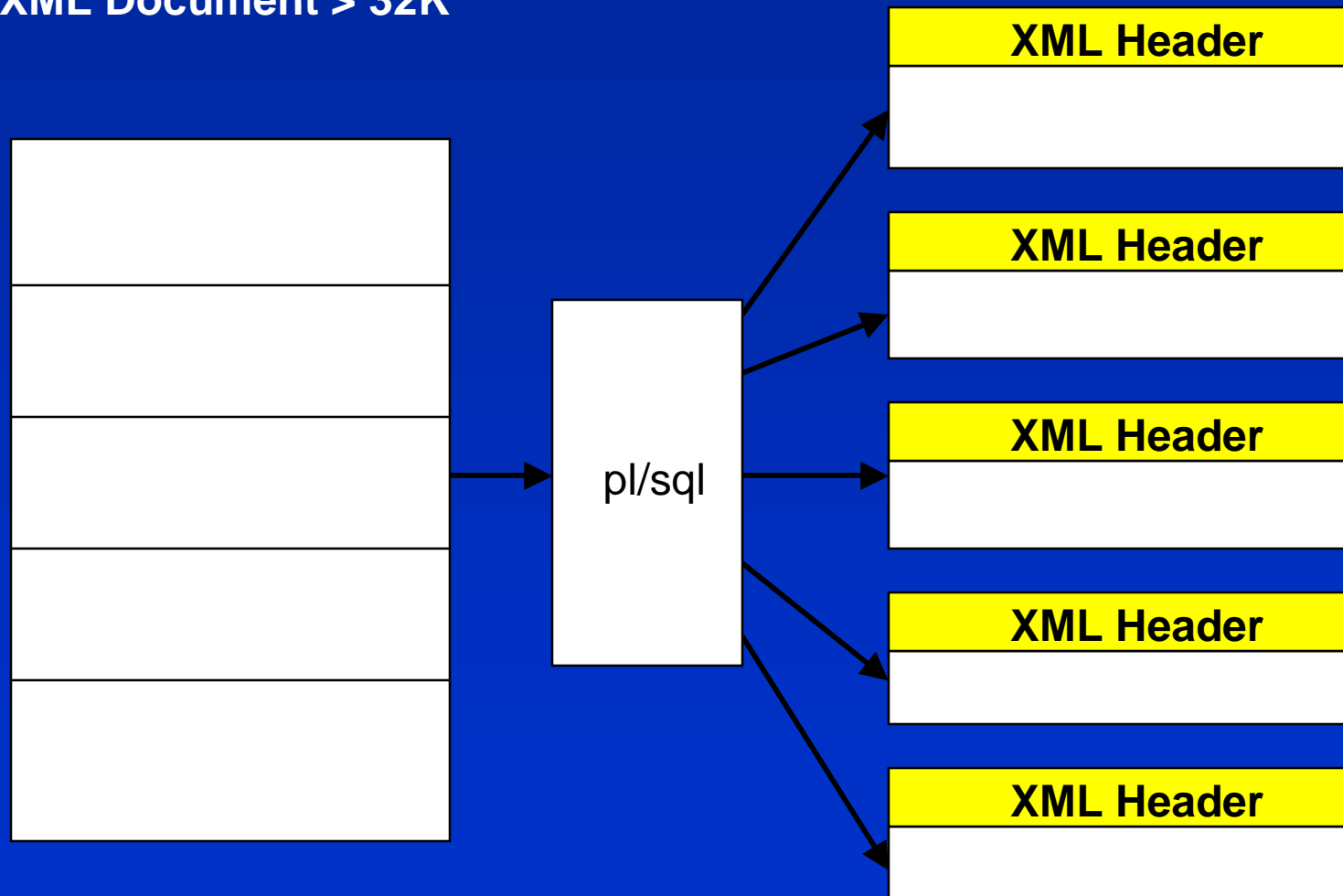
XML XPATH – parse each element, w/o table

```
-- 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">
<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;
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.
```

XML Packet Creation

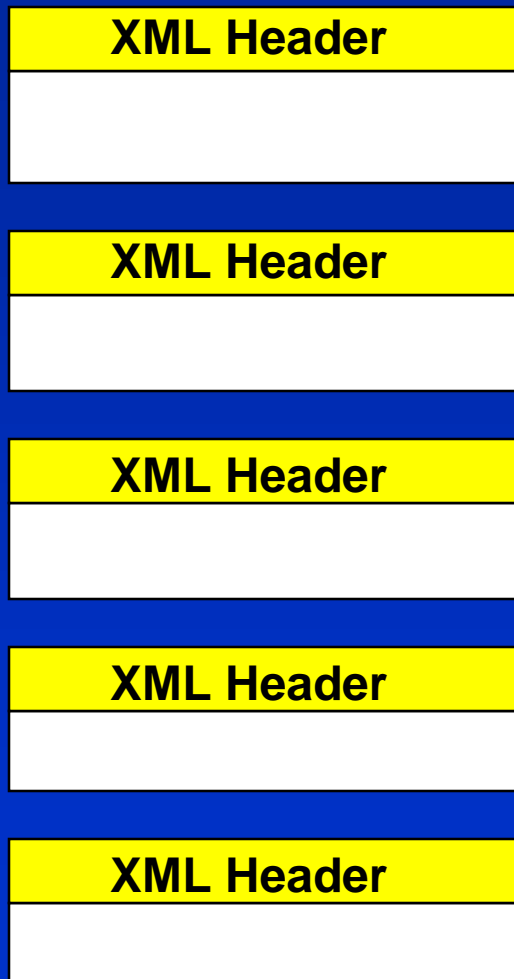
CLOB < 32 K

XML Document > 32K



XML Document Construction

CLOB < 32 K



XML Document > 32K



References

Design Tools: XMLSPY – 30 day free trial

<http://www.altova.com/simpliedownload1.html?gclid=CJuhrey1m40CFRGsGgodPwqM5w>

Oracle XML DB Docs: full implementation plus extensions of XML

<http://www.oracle.com/technology/tech/xml/xmlldb/index.html>

Oracle's SQL 10g Doc

http://oraclesvca2.oracle.com/docs/cd/B14117_01/server.101/b10759/functions204.htm

Oracle Database 10g XML & SQL – Oracle Press

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

http://www.oradev.com/xml_functions.jsp

Reference

W3.org

W3schools.com

oradev.com/xml_functions.jsp

msdn.microsoft.com/en-us/library/ms256086.aspx

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)

iso.org



Please Note

Speaker: Coleman Leviter

Integrating Oracle 10g XML: A Case Study
Part II

cleviter@ieee.org

Questions

