SDXML VT2024
Models and languages for semi-structured data and XML

Product-specific techniques Oracle Database

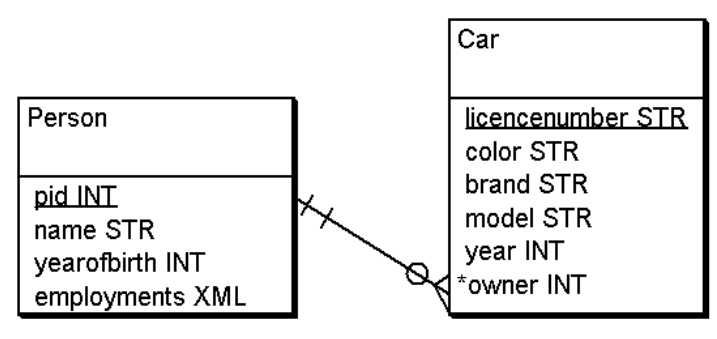
nikos dimitrakas
nikos@dsv.su.se
08-161295

Corresponding reading
Product documentation
Compendium with introduction Oracle
Section 13.3 of the course book

Oracle Database 21c

- **Support for a big part of SQL/XML according to SQL:2006**
- **Custom additions**
  - **Methods for XML objects (qualification is required!)**
    - » **Extract**
    - » **Transform**
    - » **…**
  - **Functions (many deprecated since version 12)**
    - » **Extract, ExtractValue, existsNode**
    - » **UpdateXML, InsertXML, DeleteXML, …**
    - » **XMLTransform, XMLColAttVal**
    - » **…**
  - **XQuery additions (deprecated since version 12)**
    - » **ora:view, ora:contains, ora:matches …**
- **Even more Oracle-specific solutions in previous versions, now removed or deprecated.**

Sample data



PERSON

pid	name	yearofbirth
1	John Higgins	1975
2	Steven Hendry	1973
3	Matthew Stevens	1982
4	Ronnie O´Sullivan	1980
5	Ken Doherty	1974
6	Steve Davis	1960
7	Paul Hunter	1983
8	Neil Robertson	1982

CAR

licencenumber	color	brand	model	year	owner
ABC123	black	NISSAN	Cherry	1995	1
CCD457	blue	FIAT	Forza	2001	2
DKL998	green	SAAB	9000C	1998	3
RSQ199	black	NISSAN	Micra	1999	4
WID387	red	FIAT	Nova	2003	5
ROO197	blue	SAAB	900i	1982	3
TYD226	black	NISSAN	Cherry	1990	1
PTF357	red	VOLVO	V70	2001	6
DAVIS1	red	VOLVO	V90	2007	6

Sample data

pid	employments
1	<root><employment startdate="2001-08-20" enddate="2009-02-28" employer="ABB"/> <employment startdate="2009-04-15" employer="UPC"/></root>
2	<root><employment startdate="2002-08-20" enddate="2003-06-30" employer="ABB"/> <employment startdate="2003-08-01" employer="UPC"/> <employment startdate="2006-11-01" employer="ABB"/></root>
3	<root><employment startdate="2003-01-10" employer="UPC"/> </root>
4	<root><employment startdate="2002-03-10" enddate="2010-05-22" employer="LKP"/> <employment startdate="2010-08-15" employer="STG"/></root>
5	<root><employment startdate="2002-02-12" enddate="2003-05-11" employer="LKP"/> <employment startdate="2003-05-12" enddate="2003-12-02" employer="ABB"/> <employment startdate="2003-12-06" enddate="2005-02-17" employer="LKP"/> <employment startdate="2005-02-18" enddate="2008-05-16" employer="FFD"/> <employment startdate="2008-06-02" employer="STG"/></root>
6	<root><employment startdate="2001-01-05" enddate="2005-12-31" employer="ABB"/> <employment startdate="2006-01-15" enddate="2009-01-22" employer="LKP"/> <employment startdate="2009-02-01" employer="FFD"/> <employment startdate="2009-02-01" employer="XAB"/></root>
7	<root><employment startdate="2004-01-10" enddate="2008-09-29" employer="FFD"/> <employment startdate="2008-10-01" enddate="2010-11-20" employer="LKP"/></root>
8	<root><employment startdate="2006-02-03" enddate="2008-10-30" employer="UPC"/> <employment startdate="2008-11-20" employer="ABB"/></root>

Oracle - data type

- XMLTYPE
  - Supports explicit association to XML Schema
  - Structural validation
  - Stores the validation status
  - Constructor function XMLTYPE
    - » XMLTYPE('<a/>')
    - » Only XML documents
  - Only XML document in columns
- Full validation with
  - procedure SchemaValidate
    - » stores the result
    - » can be tested with the method IsSchemaValidated
  - method IsSchemaValid
    - » returns the result
- Attribute nodes are not handled very well

Oracle - SQL/XML

- Supports the following functions
  - XMLELEMENT (partially)
  - XMLATTRIBUTES
  - XMLFOREST
  - XMLCONCAT
  - XMLCOMMENT
  - XMLPI
  - XMLAGG
  - XMLQUERY (only CONTENT?)
  - XMLTABLE (partially)
  - XMLEXISTS
  - XMLSERIALIZE
  - XMLCAST
  - XMLPARSE
  - XMLNAMESPACES is not supported, but
    - namespaces can be created with XMLATTRIBUTES!
  - Dynamic node names with the keyword EVALNAME
    - Works with XMLELEMENT, XMLATTRIBUTES, XMLFOREST, XMLPI

Oracle - other functions

- XMLCDATA
- XMLISVALID
- XMLCOLATTVAL
- XMLTRANSFORM

Deprecated:

- EXTRACT
- EXTRACTVALUE
- EXISTSNODE
- XMLSEQUENCE

- Deprecated:
  - UPDATEXML
  - APPENDCHILDXML
  - INSERTCHILDXML
  - INSERTCHILDXMLAFTER
  - INSERTCHILDXMLBEFORE
  - INSERTXMLAFTER
  - INSERTXMLBEFORE
  - DELETEXML

- XMLROOT

Oracle - XMLTYPE

- Methods (sometimes referred to as "member functions")
  - extract
  - existsNode
  - transform
  - isSchemaValidated
  - isSchemaValid
  - isSchemaBased
  - isFragment
  - getStringVal
  - getNumberVal
  - getCLOBVal
  - getBLOBVal
  - getNamespace
  - getRootElement
  - getSchemaURL

Oracle - XMLCDATA

- Equivalent to SQL/XML's XMLTEXT
  - Puts the text inside <![CDATA[value]]>

```
SELECT XMLELEMENT(NAME Person, XMLCDATA(name))
FROM person
WHERE pid = 1
```

```
<PERSON><![CDATA[John Higgins]]></PERSON>
```

```
SELECT XMLELEMENT(NAME Sign, '<') FROM dual
<SIGN>&lt;</SIGN>

SELECT XMLELEMENT(NAME Sign, XMLCDATA('<')) FROM dual
<SIGN><![CDATA[<]]></SIGN>
```


Oracle - XMLISVALID

- Equivalent to SQL/XML's XMLVALIDATE
  - implicit schema
    XMLISVALID(xml-value)
  - explicit schema
    XMLISVALID(xml-value, schema-name)

Oracle - XMLCOLATTVAL

- Generates one XML fragment per row
  - One column element per column/cell

```
SELECT XMLCOLATTVAL(name, yearofbirth)
FROM person WHERE pid = 1
```

```
<column name = "NAME">John Higgins</column>
<column name = "YEAROFBIRTH">1975</column>
```

```
SELECT XMLCOLATTVAL('nikos' AS "Subject", 'lectures' AS "Predicate")
FROM dual
```

```
<column name = "Subject">nikos</column>
<column name = "Predicate">lectures</column>
```


Oracle - EXTRACT

- Applies an XPath expression to an XML value
  - Returns XML
  - Deprecated. Replaced by SQL/XML's XMLQUERY

```
SELECT name, EXTRACT(employments, '//employment[1]')
FROM Person
```

John Higgins	<employment startdate="2001-08-20" enddate="2009-02-28" employer="ABB"/>
Stephen Hendry	<employment startdate="2002-08-20" enddate="2003-06-30" employer="ABB"/>
Matthew Stevens	<employment startdate="2003-01-10" employer="UPC"/>
Ronnie O'Sullivan	<employment startdate="2002-03-10" enddate="2010-05-22" employer="LKP"/>
Ken Doherty	<employment startdate="2002-02-12" enddate="2003-05-11" employer="LKP"/>
Steve Davis	<employment startdate="2001-01-05" enddate="2005-12-31" employer="ABB"/>
Paul Hunter	<employment startdate="2004-01-10" enddate="2008-09-29" employer="FFD"/>
Neil Robertson	<employment startdate="2006-02-03" enddate="2008-10-30" employer="UPC"/>

Oracle - EXTRACT

```
SELECT name,
    EXTRACT(employments, '//employment[1]/@employer')
FROM person
```

John Higgins	ABB
Stephen Hendry	ABB
Matthew Stevens	UPC
Ronnie O'Sullivan	LKP
Ken Doherty	LKP
Steve Davis	ABB
Paul Hunter	FFD
Neil Robertson	UPC

Oracle - EXTRACTVALUE

- Applies an XPath expression to an XML value
  - Returns a value
  - Deprecated. Replaced by SQL/XML's XMLQUERY

```
SELECT name,
    EXTRACTVALUE(employments, '//employment[1]/@employer')
FROM Person
```

John Higgins	ABB
Stephen Hendry	ABB
Matthew Stevens	UPC
Ronnie O'Sullivan	LKP
Ken Doherty	LKP
Steve Davis	ABB
Paul Hunter	FFD
Neil Robertson	UPC

Oracle - EXISTSNODE

- Tests if an XPath expression matches at least one node in an XML value
  - Deprecated. Replaced by SQL/XML's XMLEXISTS
  - Returns 1 (true) or 0 (false)

```
SELECT name
FROM Person
WHERE EXISTSNODE(employments,
'//employment[@employer="ABB"]') = 1
```

```
John Higgins
Stephen Hendry
Ken Doherty
Steve Davis
Neil Robertson
```


Oracle - XMLSEQUENCE

- Splits an XML fragment/sequence into many XML values
  - Deprecated. Replaced by SQL/XML's XMLTABLE

```
SELECT e.*
FROM Person,
  TABLE(XMLSEQUENCE(EXTRACT(employments, '//employment'))) e
WHERE pid = 5
```

```
<employment startdate="2002-02-12" enddate="2003-05-11" employer="LKP"/>
<employment startdate="2003-05-12" enddate="2003-12-02" employer="ABB"/>
<employment startdate="2003-12-06" enddate="2005-02-17" employer="LKP"/>
<employment startdate="2005-02-18" enddate="2008-05-16" employer="FFD"/>
<employment startdate="2008-06-02" employer="STG"/>
```

Five rows in the result.

Oracle - XMLTRANSFORM

- Transforms an XML document according to an XSLT document (XSLT 1.0)

```
SELECT XMLTRANSFORM(employments ,
'<xsl:transform xmlns:xsl="http://www.w3.org/1999/XSL/Transform" version="1.0">
  <xsl:output method="xml"/>
  <xsl:template match="/">
    <xsl:element name="Employers">
      <xsl:for-each select="//employment/@employer">
       <xsl:element name="Employer"><xsl:value-of select="."/></xsl:element>
      </xsl:for-each>
    </xsl:element>
  </xsl:template>
</xsl:transform>')
FROM Person
WHERE name = 'Ken Doherty'
```

```
<?xml version="1.0" encoding="UTF-8"?>
<Employers>
  <Employer>LKP</Employer>
  <Employer>ABB</Employer>
  <Employer>LKP</Employer>
  <Employer>FFD</Employer>
  <Employer>STG</Employer>
</Employers>
```


Oracle - XMLTRANSFORM

```
SELECT XMLTRANSFORM(employments ,
'<xsl:transform xmlns:xsl="http://www.w3.org/1999/XSL/Transform" version="1.0">
  <xsl:output method="xml"/>
  <xsl:template match="/">
    <xsl:element name="Employers">
      <xsl:for-each select="//employment[not (@employer =
                                    preceding::employment/@employer)]/@employer">
       <xsl:element name="Employer">
        <xsl:value-of select="."/>
       </xsl:element>
      </xsl:for-each>
    </xsl:element>
  </xsl:template>
</xsl:transform>')
FROM Person
WHERE name = 'Ken Doherty'
```

```
<?xml version="1.0" encoding="UTF-8"?>
<Employers>
  <Employer>LKP</Employer>
  <Employer>ABB</Employer>
  <Employer>FFD</Employer>
  <Employer>STG</Employer>
</Employers>
```

Oracle - XMLTYPE DML

- DEPRECATED
  Functions that change an XML value and return the modified XML value
  - UpdateXML
  - DeleteXML
  - AppendXML
  - InsertChildXML, InsertChildXMLBefore, InsertChildXMLAfter
  - InsertXMLBefore, InsertXMLAfter
- The whole XML value in a cell must be replaced

```
UPDATE Person
SET employments = DML-function(employments, …)
WHERE …
```


Oracle - XML DML

- XQuery Update Facility
  - some syntax differences
- transform statements
  - copy clause
  - modify clause
    - » delete node(s)
    - » insert node(s)
    - » rename node
    - » replace node
  - return clause
  - keywords for insert
    - » before
    - » after
    - » as first into
    - » as last into
    - » into
  - keywords for replace
    - » (value of) … with …
  - keywords for rename
    - » as

Oracle - XML DML

- The whole XML value in a cell must be replaced

```
UPDATE Person
SET employments = XMLQUERY('transform statement'
                           PASSING employments
                           RETURNING CONTENT)
WHERE …
```


Oracle - transform - insert

```
XQUERY
copy $x := <root><a>456</a><a>789</a></root>
modify insert node <a>123</a> as first into $x
return $x
```

```
<root>
    <a>123</a>
    <a>456</a>
    <a>789</a>
</root>
```

Oracle - transform - insert

```
XQUERY
copy $x := <root><a>456</a><a>789</a></root>
modify insert node <a>123</a> before $x/a[2]
return $x

<root>
        <a>456</a>
        <a>123</a>
        <a>789</a>
</root>
```

Buggy in versions prior to 18:
Didn't work: [text() = 789] or [. = 789]
Worked: [number(.) = 789] or [string(.) = "789"]

Oracle - transform - insert

```
XQUERY
copy $x := <root><a>456</a><a>789</a></root>
modify insert node attribute c {5} into $x/a[2]
return $x

<root>
        <a>456</a>
        <a c="5">789</a>
</root>
```

Oracle - transform - insert

```
UPDATE Person
SET employments =  XMLQUERY('
        copy $ne := $e
        modify insert node element employment {
                        attribute startdate {"2011-09-01"},
                        attribute employer {"LBM"}} as last into $ne/root
        return $ne' PASSING employments AS "e"
                RETURNING CONTENT)
WHERE pid = 3


SELECT employments FROM Person WHERE pid = 3


<root>
    <employment startdate="2003-01-10" employer="UPC"/>
    <employment startdate="2011-09-01" employer="LBM"/>
</root>
```


Oracle - transform - delete

```
XQUERY
copy $x := <root><a>456</a><a>789</a></root>
modify delete node $x/a[2]
return $x


<root>
        <a>456</a>
</root>
```

Buggy in versions prior to 18:

Didn't work: [text() = 789] or [. = 789]

Worked: [number(.) = 789] or [string(.) = "789"]

Oracle - transform - delete

```
SELECT XMLQUERY('copy $ne := $e
                 modify delete node $ne//employment[2]
                 return $ne' PASSING employments AS "e"
                 RETURNING CONTENT)
FROM Person
WHERE pid = 3
```

```
<root>
    <employment startdate="2003-01-10" employer="UPC"/>
</root>
```

```
SELECT employments FROM person WHERE pid = 3
```

```
<root>
    <employment startdate="2003-01-10" employer="UPC"/>
    <employment startdate="2011-09-01" employer="LBM"/>
</root>
```


Oracle - transform - delete

```
UPDATE Person
SET employments = XMLQUERY('copy $ne := $e
                            modify delete node $ne//employment[2]
                            return $ne' PASSING employments AS "e"
                                    RETURNING CONTENT)
WHERE pid = 3
```

```
SELECT employments FROM Person WHERE pid = 3
```

```
<root>
    <employment startdate="2003-01-10" employer="UPC"/>
</root>
```

Oracle - transform - replace

```
XQUERY
copy $x := <root><a>456</a><a>789</a></root>
modify replace node $x/a[2] with <b>123</b>
return $x
```

```
<root>
        <a>456</a>
        <b>123</b>
</root>
```

Buggy in versions prior to 18:
Didn't work: [text() = 789] or [. = 789]
Worked: [number(.) = 789] or [string(.) = "789"]

Oracle - transform - replace

```
XQUERY
copy $x := <root><a>456</a><a>789</a></root>
modify replace value of node $x/a[2] with 123
return $x
```

```
<root>
        <a>456</a>
        <a>123</a>
</root>
```

Oracle - transform - replace

```
XQUERY
copy $x := <root><a b="ccc">456</a><a>789</a></root>
modify replace node $x/a[1]/@b with attribute f {"ddd"}
return $x
```

```
<root>
        <a f="ddd">456</a>
        <a>789</a>
</root>
```


Oracle - transform - rename

```
XQUERY
copy $x := <root><a>456</a><a>789</a></root>
modify rename node $x/a[2] as "b"
return $x
```

```
<root>
        <a>456</a>
        <b>789</b>
</root>
```

Oracle - transform - rename

```
XQUERY
copy $x := <root><a>456</a><a>789</a></root>
modify for $a in $x/a
        return rename node $a as "b"
return $x
```

Does not work. Returns $x unchanged:

```
<root>
        <a>456</a>
        <a>789</a>
</root>
```


Oracle - transform - rename

But this works:

```
XQUERY
copy $x := <root><a>456</a><a>789</a></root>
modify (rename node $x/a[1] as "b",
            rename node $x/a[2] as "b")
return $x
```

Result:

```
<root>
        <b>456</b>
        <b>789</b>
</root>
```

Oracle - transform - rename

But this does not work:

```
XQUERY
copy $x := <root><a>456</a><a>789</a></root>
modify for $i in 1 to count($x/a)
        return rename node $x/a[position() = $i] as "b"
return $x
```

The result is unchanged:

```
<root>
        <a>456</a>
        <a>789</a>
</root>
```


Oracle - XMLTYPE methods

- For XML objects in a column, the column name must be qualified:
  - table-alias.column.method()
- Most methods correspond to functions where the XML object is the first argument. Example:
  - EXTRACT(xml-object, xpath-expression)
  - xml-object.extract(xpath-expression)

Oracle - extract

- Corresponds to the function EXTRACT
- Returns XML based on an XPath expression

```
SELECT EXTRACT(employments, '//employment[1]'),
    p.employments.extract('//employment[1]')
FROM Person p
```

The two columns in the result are identical.

Oracle - existsNode

- Corresponds to the function EXISTSNODE
- Checks if an XPath expression matches at least one node
  - Returns 1 (true) or 0 (false)

```
SELECT name,
    EXISTSNODE(employments, '//employment[@employer="ABB"]'),
    p.employments.existsNode('//employment[@employer="ABB"]')
FROM Person p
```

John Higgins	1	1
Stephen Hendry	1	1
Matthew Stevens	0	0
Ronnie O'Sullivan	0	0
Ken Doherty	1	1
Steve Davis	1	1
Paul Hunter	0	0
Neil Robertson	1	1

Oracle - transform

- Corresponds to the function XMLTRANSFORM
  - Does not accept the XSLT document as a string

```
SELECT p.employments.transform(XMLTYPE('
<xsl:transform xmlns:xsl="http://www.w3.org/1999/XSL/Transform" version="1.0">
  <xsl:output method="xml"/>
  <xsl:template match="/">
    <xsl:element name="Employers">
      <xsl:for-each select="//employment/@employer">
       <xsl:element name="Employer"><xsl:value-of select="."/></xsl:element>
      </xsl:for-each>
    </xsl:element>
  </xsl:template>
</xsl:transform>'))
FROM Person p
WHERE name = 'Stephen Hendry'
```

```
<Employers>
  <Employer>ABB</Employer>
  <Employer>UPC</Employer>
  <Employer>ABB</Employer>
</Employers>
```


Oracle - get…Val

- Methods for converting to other data types
  - getNumberVal
  - getStringVal Deprecated. Replaced by SQL/XML:s XMLSERIALIZE
  - getBLOBVal Deprecated. Replaced by SQL/XML:s XMLSERIALIZE
  - getCLOBVal Deprecated. Replaced by SQL/XML:s XMLSERIALIZE
- Return the value of the node or the node as a value of the corresponding type
  - getNumberVal requires that the node's value is compatible.
  - Behave differently with attribute nodes and element nodes or text nodes.

```
SELECT XMLTYPE('<a v="44">55</a>').extract('//@v').getNumberVal()
FROM dual
```

```
SELECT XMLTYPE('<a v="44">55</a>').extract('//text()').getNumberVal()
FROM dual
```

Oracle - getRootElement

- Returns the name of the root element
  - NULL if the XML object is a fragment
  - Deprecated. Replaced by the XQuery function local-name

```
SELECT XMLTYPE('<TheRoot />').getRootElement()
FROM dual
```

Returns "TheRoot"

```
SELECT EXTRACT(XMLTYPE('<a><b/><b/></a>'),'//b').getRootElement()
FROM dual
```

or

```
SELECT XMLQUERY('for $a in (1,2) return <b/>'
                RETURNING CONTENT).getRootElement()
FROM dual
```

Return NULL

XMLTYPE('<b/><b/>') does not work!

Oracle - getSchemaURL

- Returns the URL to the XML Schema of the XML document
  - NULL if no schema is associated

```
SELECT XMLTYPE('<a />').getSchemaURL()
FROM dual
```

Returns NULL

Oracle - schema methods

- isSchemaValid
  - Corresponds to the function XMLISVALID
  - Possible to specify an XML Schema
  - Returns the result (1 or 0)
- isSchemaValidated
  - returns the stored value
- isSchemaBased
  - returns 0 or 1

Oracle - isFragment

- Checks if an XML value is a fragment or a document
  - Returns 1 (fragment) or 0 (document)

```
SELECT XMLQUERY('for $a in (1,2) return <b/>'
                                RETURNING CONTENT).isFragment()
FROM dual
```

Returns 1

```
SELECT XMLTYPE('<a><b/><b/></a>').isFragment()
FROM dual
```

Returns 0

Oracle - ora:view

- Oracle-specific XQuery function
  - takes the name of a table/view (qualified with a schema if necessary)
  - returns an XML fragment where
    - » each row is a ROW element
    - » each column is an element with the column name as the element name and the value as a text node

```
SELECT XMLQUERY('ora:view("person")' RETURNING CONTENT)
FROM dual
```

```
<ROW><PID>1</PID><NAME>John Higgins</NAME><YEAROFBIRTH>1975</YEAROFBIRTH><EMPLOYMENTS><root>
 <employment startdate="2001-08-20" enddate="2009-02-28" employer="ABB"/>
 <employment startdate="2009-04-15" employer="UPC"/>
</root>
</EMPLOYMENTS></ROW>
<ROW><PID>2</PID><NAME>Stephen Hendry</NAME><YEAROFBIRTH>1973</YEAROFBIRTH><EMPLOYMENTS><root>
 <employment startdate="2002-08-20" enddate="2003-06-30" employer="ABB"/>
 <employment startdate="2003-08-01" employer="UPC"/>
 <employment startdate="2006-11-01" employer="ABB"/>
</root>
</EMPLOYMENTS></ROW>
...
```


Oracle - ora:view

```
SELECT XMLQUERY('for $r in ora:view("person")/ROW[PID = (1,2,3)]
                 let $pd := $r/PID | $r/NAME
                 return element Person {$pd}'
                 RETURNING CONTENT)
FROM dual
```

```
<Person><PID>1</PID><NAME>John Higgins</NAME></Person>
<Person><PID>2</PID><NAME>Stephen Hendry</NAME></Person>
<Person><PID>3</PID><NAME>Matthew Stevens</NAME></Person>
```

Oracle - ora:view

- DEPRECATED. Replaced by fn:collection
  - takes a URI as parameter:
  - oradb:/schema-name/table-name
  - oradb:/PUBLIC/table-name
  - Case-sensitive

```
SELECT XMLQUERY('fn:collection("oradb:/PUBLIC/PERSON")'
     RETURNING CONTENT)
FROM dual
```

```
<ROW><PID>1</PID><NAME>John Higgins</NAME><YEAROFBIRTH>1975</YEAROFBIRTH><EMPLOYMENTS><root>
 <employment startdate="2001-08-20" enddate="2009-02-28" employer="ABB"/>
 <employment startdate="2009-04-15" employer="UPC"/>
</root>
</EMPLOYMENTS></ROW>
<ROW><PID>2</PID><NAME>Stephen Hendry</NAME><YEAROFBIRTH>1973</YEAROFBIRTH><EMPLOYMENTS><root>
 <employment startdate="2002-08-20" enddate="2003-06-30" employer="ABB"/>
 <employment startdate="2003-08-01" employer="UPC"/>
 <employment startdate="2006-11-01" employer="ABB"/>
</root>
</EMPLOYMENTS></ROW>
…
```


Oracle - XQUERY

- Support for XQuery
  - XQuery statements after the keyword XQUERY

```
XQUERY
for $a in (1,2,3)
return element Number {$a}
```

Three rows in the result:

```
<Number>1</Number>
<Number>2</Number>
<Number>3</Number>
```

```
XQUERY
element Result  {for $a in (1,2,3)
                 return element Number {$a}}
```

```
<Result>
    <Number>1</Number><Number>2</Number><Number>3</Number>
</Result>
```

Oracle Database & JSON

- JSON data type
- Functions (SQL/JSON)
  - JSON_VALUE, JSON_QUERY, JSON_TABLE, JSON_EXISTS
  - JSON_ARRAYAGG, JSON_OBJECTAGG

Summary

- Oracle follows the SQL standard for the most part
- No XQuery 3
- Not all functions are supported
- Weird handling of attribute nodes, node values
- Nested loops in transform statements don't work

- Use of standard SQL makes migration easier
  - Avoid product specific solutions when possible
  - Avoid deprecated solutions when possible

What to do next

- Introduction to Oracle & XML (compendium)
  - Introduction to Oracle and SQL Developer
  - Examples
- Assignment 7 (Oracle & XML)