ORACLE®

An Introduction to Oracle XML DB in Oracle Database 12c Release 2

Mark D Drake
Manager, Product Management

Copyright © 2017, Oracle and/or its affiliates. All rights reserved. |

The following is intended to outline our general product direction. It is intended for information purposes only, and may not be incorporated into any contract. It is not a commitment to deliver any material, code, or functionality, and should not be relied upon in making purchasing decisions. The development, release, and timing of any features or functionality described for Oracle's products remains at the sole discretion of Oracle.

Copyright © 2012, Oracle and/or its affiliates. All rights reserved.
Insert Information Protection Policy Classification from Slide 8

Agenda

1. Introduction to XML DB
2. XMLType and XQuery
3. Loading XML
4. XML Generation
5. XML and SQL Integration

Copyright © 2017, Oracle and/or its affiliates. All rights reserved. |

Agenda

6 XML Storage and Indexing

7 XML Schema

8 XML DB Repository

9 Database Native Web Services

10 Summary

Copyright © 2017, Oracle and/or its affiliates. All rights reserved. |

Introduction to XML DB

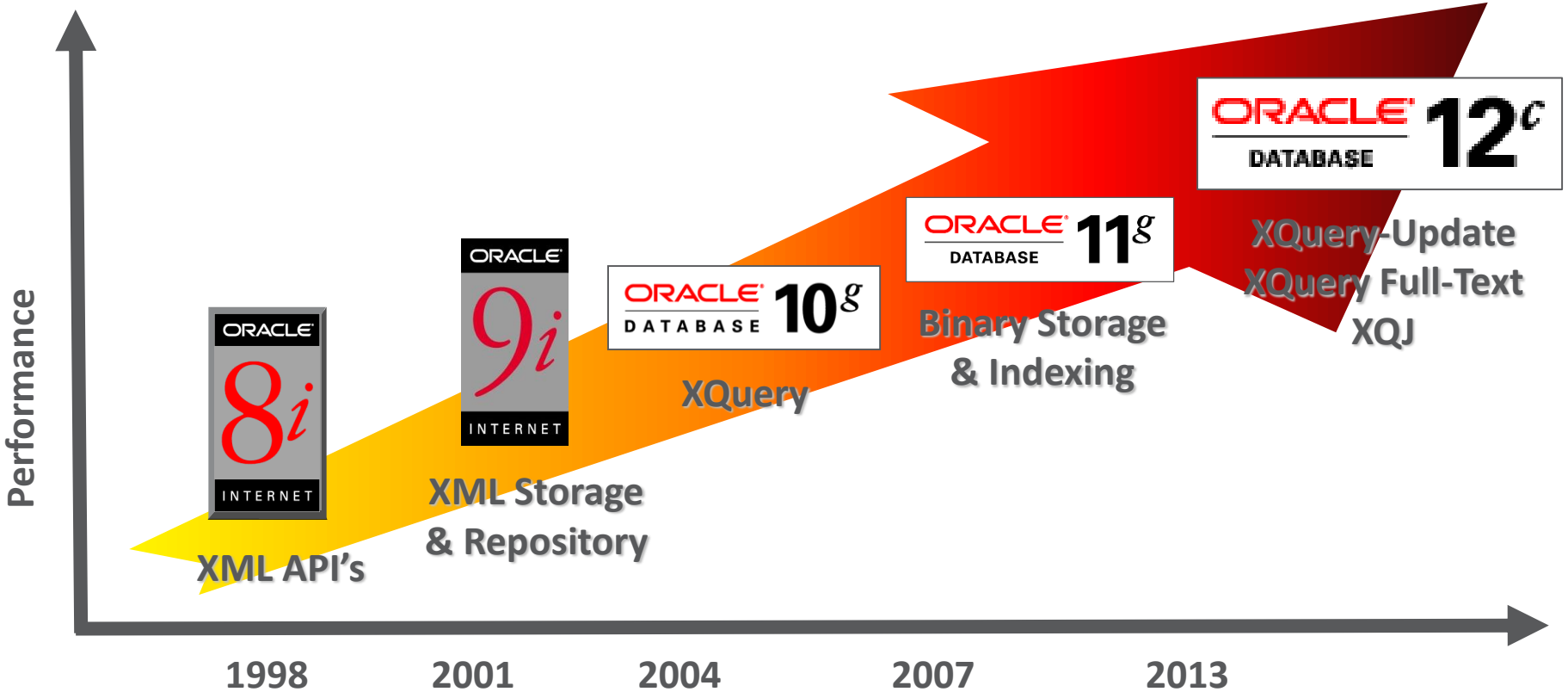
Copyright © 2017, Oracle and/or its affiliates. All rights reserved. |

Oracle's XML Vision

- Enable a single source of truth for XML
- Provide the best platform for managing all your XML
  - Flexibility to allow optimal processing of data-centric and content-centric XML
  - Deliver Oracle's commitment to Reliability, Security, Availability and Scalability
- Drive and implement key XML Standards
- Support SQL-centric, XML-centric and document-centric development paradigms
- Support XML in Database and Application Server

Copyright © 2017, Oracle and/or its affiliates. All rights reserved. |

Oracle & XML : Sustained Innovation



Copyright © 2017, Oracle and/or its affiliates. All rights reserved. |

XMLType and XQuery

Copyright © 2017, Oracle and/or its affiliates. All rights reserved. |

XMLType

- Standard data type, makes database XML aware
  - Use as Table, Column, Variable, Argument or Return Value
- Abstraction for managing XML
  - Enforces XML content model and XML fidelity
  - Enables XML schema validation
  - Multiple persistence and indexing options
- Query and update operations performed using XQuery
- All application logic is independent of persistence model

ORACLE

Copyright © 2017, Oracle and/or its affiliates. All rights reserved. |

Using XMLType

```
create table INVOICES of XMLTYPE;
```

```
create table PURCHCASEORDERS (
  PO_NUMBER NUMBER(4),
  PO_DETAILS XMLTYPE
)
XMLTYPE column PO_DETAILS
XMLSCHEMA "http://schemas.example.com/PurchaseOrder.xsd"
ELEMENT “PurchaseOrder“
STORE AS OBJECT RELATIONAL;
```

Copyright © 2017, Oracle and/or its affiliates. All rights reserved. |

XQuery

- W3C standard for generating, querying and updating XML
  - Natural query language for XML content
  - Evolved from XPath and XSLT
  - Analogous to SQL in the relational world
- Iterative rather than Set-Based
- Basic construct is the FLWOR clause
  - for, let where, order by, return
- XQuery operations result in a sequence consisting of zero or more nodes

ORACLE

Copyright © 2017, Oracle and/or its affiliates. All rights reserved. |

XQuery FLWOR example

```
<PurchaseOrder   DateCreated=“2011-01-31”>
   …
  <LineItems>
    <LineItem ItemNumber="1">
      <Part Description="Octopus“>31398750123</Part>
      <Quantity>3.0</Quantity>
    </LineItem>
…..
    <LineItem ItemNumber="5">
      <Part Description="King Ralph">18713810168</Part>
      <Quantity>7.0</Quantity>
    </LineItem>
  </LineItems>
</PurchaseOrder>
```

for $l in $PO/PurchaseOrder/LineItems/LineItem
return $l/Part/@Description

Octopus
….

King Ralph

Copyright © 2017, Oracle and/or its affiliates. All rights reserved. |

XQuery fn:collection : Working with lots of XML

```
for $doc in fn:collection("oradb:/OE/PURCHASEORDER")
    return $doc
```

- Used to access a collection of documents
  - Allows an XQuery to operate on a set of XML documents
- Collection sources include
  - The contents of a folder
  - XMLType tables or columns
  - Relational Tables via a conical mapping scheme
- Protocol "oradb:" causes the components of the path should be interpreted as a Schema, Table, Column
  - Column is optional

ORACLE

Copyright © 2017, Oracle and/or its affiliates. All rights reserved. |

XQuery : Where and Order by clause

```
let $USER := “SKING”
for $doc in fn:collection(“oradb:/OE/PURCHASEORDER”)
    where $doc/PurchaseOrder[User = $USER]
    order by $doc/PurchaseOrder/Reference
    return $doc/PurchaseOrder/Reference
```

- Where clause controls which documents or nodes are processed
  - Enables the use of predicates
- Order by clause controls the ordering of nodes

Copyright © 2017, Oracle and/or its affiliates. All rights reserved. |

Executing XQuery in SQL*Plus

```
SQL> XQUERY
  2  let $USER := "SKING"
  3   for $doc in fn:collection("oradb:/OE/PURCHASEORDER")
  4     where $doc/PurchaseOrder[User = $USER]
  5     order by $doc/PurchaseOrder/Reference
  6     return $doc/PurchaseOrder/Reference
  7  /
```

- If XQuery statement ends with ‘;’ use empty comment (: :) to prevent semi-colon being interpreted by SQL.

Copyright © 2017, Oracle and/or its affiliates. All rights reserved. |

Executing XQuery using XMLTable

```
select *
 from XMLTABLE
    (
     'for $doc in fn:collection("oradb:/OE/PURCHASEORDER")
                  return $doc/PurchaseOrder/Reference'
    )
```

- Converts the sequence returned by XQuery into a relational result set
- JDBC / OCI programs
- Tools that do not yet provide native XQuery support
  - SQL*Developer, APEX SQL Workbench
- This is what the SQL*PLUS XQUERY command does under the covers

Copyright © 2017, Oracle and/or its affiliates. All rights reserved. |

XQuery in Database Native Web Services

```
<ENV:Envelope
        xmlns:ENV="http://schemas.xmlsoap.org/soap/envelope/"
        xmlns:ENC="http://schemas.xmlsoap.org/soap/encoding/"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xmlns:xsd="http://www.w3.org/2001/XMLSchema">
    <ENV:Body>
       <m:query xmlns:m="http://xmlns.oracle.com/orawsv">
                    <m:query_text type="XQUERY“> for $doc in fn:collection("oradb:/OE/PURCHASEORDER")
                     return $doc/PurchaseOrder/Reference </m:query_text>
       <m:pretty_print>true</m:pretty_print>
      </m:query>
   </ENV:Body>
</ENV:Envelope>
```

- WSDL location : http://dbserver:port/orawsv?wsdl

Copyright © 2017, Oracle and/or its affiliates. All rights reserved. |

JCR-225 or XQJ

```
import javax.xml.xquery.*
XQDataSource dataSource = new oracle.xml.xquery.OXQDataSource();
XQConnection connection = dataSource.getConnection();
XQExpression expression = connection.createExpression();
XQResultSequence result = expression.executeQuery ("for $doc in fn:collection(\"oradb:/OE/PURCHASEORDER\")  return
$doc/PurchaseOrder/Reference");
result.writeSequence(System.out, null);
result.close();
```

- Native XQuery API for Java
- XQJ is to XQuery what JDBC is to SQL
- Reference implementation by Oracle XMLDB

Copyright © 2017, Oracle and/or its affiliates. All rights reserved. |

Loading XML

Copyright © 2017, Oracle and/or its affiliates. All rights reserved. |

SQL Loader

```
load data
infile 'filelist.dat'
append
into table PURCHASEORDER
xmltype(XMLDATA)
(
 filename filler char(120),
 XMLDATA  lobfile(filename) terminated by eof
)
```

```
C:\purchaseOrders\ABANDA-20020405224101614PST.xml
C:\purchaseOrders\ABANDA-20020406224701221PDT.xml
…
```

- Load files from a local file system
- Filelist.dat contains a list of the files to be loaded

Copyright © 2017, Oracle and/or its affiliates. All rights reserved. |

BFILE Constructor

```
create or replace directory XMLDIR as ‘c:\myxmlfiles’;

insert into PURCHASEORDER values (
  XMLTYPE (
    BFILENAME(‘XMLDIR’, ‘SKING-20021009123335560PDT.xml’),
    NLS_CHARSET_ID(‘AL32UTF8’)));
```

- Directory XMLDIR references a directory in a file system local to the database server
- Must specify the character set encoding of the file being loaded.

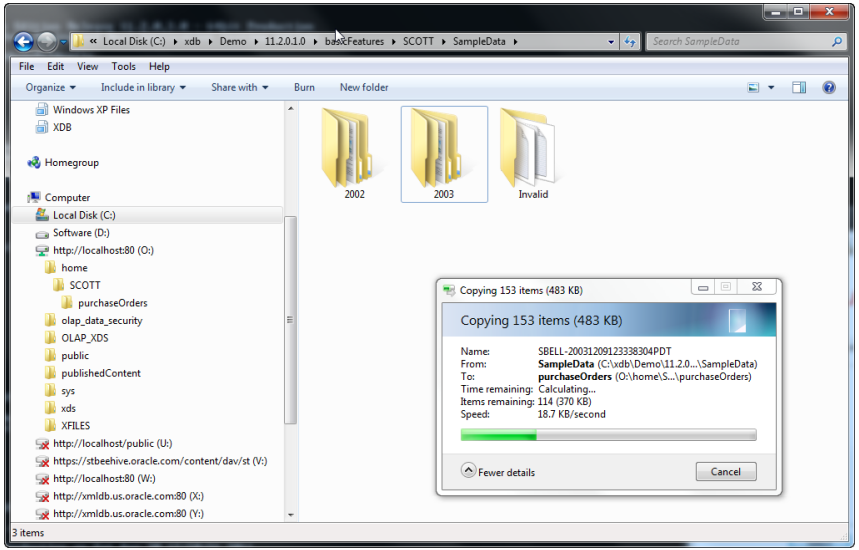
Copyright © 2017, Oracle and/or its affiliates. All rights reserved. |

JDBC, OCI, PL/SQL

```
public boolean doInsert(String filename) throws SQLException,
                                                FileNotFoundException {

    String statementText = "insert into PURCHASEORDER values (:1)“;
    Connection conn = getConnection();
    OracleCallableStatement statement =
        (OracleCallableStatement) conn.prepareStatement(statementText);

    FileInputStream is = new FileInputStream(filename);
    XMLType xml = XMLType.createXML(this.getConnection(),is);
    statement.setObject(1,xml);
    boolean result = statement.execute();

    statement.close();
    conn.commit();
    return result;
 }
```

Copyright © 2017, Oracle and/or its affiliates. All rights reserved. |

The XML DB repository and protocols



- Allows content to be loaded directly into the database using FTP, HTTP and WebDAV

Copyright © 2017, Oracle and/or its affiliates. All rights reserved. |

XML Generation

Copyright © 2017, Oracle and/or its affiliates. All rights reserved. |

Generating XML from relational data

- SQL/XML makes it easy to generate XML from relational data
  - Result set generated by SQL Query consists of one or more XML documents
- XQuery enables template-based generation of XML from relational tables
  - Use fn:collection() to generate a canonical XML representation of relational data that can be transformed using XQuery
- XMLType views enable persistent XML centric access to relational content

ORACLE

Copyright © 2017, Oracle and/or its affiliates. All rights reserved. |

Generating XML using SQL/XML

- XMLElement()
  - Generates an Element with simple or complex content
    - Simple Content derived from a scalar column or constant
    - Complex content derived from XMLType columns or generated using nested SQL/XML operators
- XMLAttributes()
  - Adds attributes to an element
- XMLAgg()
  - Generates an XML Fragment
  - Aggregation operator used to process the results of a nested sub-query

ORACLE

Copyright © 2017, Oracle and/or its affiliates. All rights reserved. |

Example : Using SQL/XML

```
select  xmlElement ( "Department",
            xmlAttributes( d.DEPTNO as “Id"),
            xmlElement("Name", d.DNAME),
            xmlElement("Employees”, (
                 select xmlAgg(
                              xmlElement("Employee",
                                   xmlForest(
                                        e.ENAME as "Name",
                                        e.HIREDATE as"StartDate”
                                    )
                               )
                          )
                    from EMP e
                   where e.DEPTNO = d.DEPTNO
            ) )
        ) as XML
  from DEPT d
```

XML

```
<Department Id="10">
 <Name>ACCOUNTING</Name>
 <Employees>
  <Employee employeeId="7782">
   <Name>CLARK</Name>
   <StartDate>1981-06-09</StartDate>
  </Employee>
  <Employee”>
   <Name>KING</Name>
   <StartDate>1981-11-17</StartDate>
  </Employee>
  <Employee>
   <Name>MILLER</Name>
   <StartDate>1982-01-23</StartDate>
  </Employee>
 </Employees>
</Department>
```

Copyright © 2017, Oracle and/or its affiliates. All rights reserved. |

XML and SQL Integration

Copyright © 2017, Oracle and/or its affiliates. All rights reserved. |

XMLExists() XQuery Predicates

```
SQL> select OBJECT_VALUE  “XML”
 2       from PURCHASEORDER
 3     where XMLEXISTS (
 4                    '$PO/PurchaseOrder[Reference=$REF]’
 5                     passing OBJECT_VALUE as "PO",
 6                     'SKING-20021009123336131PDT' as "REF“
 7                );
```

XML

```
<PurchaseOrder >
 <Reference>SKING-20021009123336131PDT</Reference>
 …
</PurchaseOrder >
```

- Use in SQL where clause to filter rows based on an XQuery expression
- Bind variables are supplied via the “Passing” clause

Copyright © 2017, Oracle and/or its affiliates. All rights reserved. |

XMLQuery() Fragment access

```
SQL> select XMLQUERY(
  2                  '$PO/PurchaseOrder/ShippingInstructions'
  3                  passing OBJECT_VALUE as "PO"
  4                  returning content) XML
  5     from PURCHASEORDER
  6    where XMLEXISTS(
  7                  '$PO/PurchaseOrder[Reference=$REF]'
  8                  passing OBJECT_VALUE as "PO",  'SKING-20021009123336131PDT' as "REF");
```

XML

```
<ShippingInstructions>
 <name>Steven A. King</name>
...
</ShippingInstructions>
```

- Use in SQL where clause to extract a fragment from each document in a result set.
- Bind variables are supplied via the “Passing” clause

Copyright © 2017, Oracle and/or its affiliates. All rights reserved. |

XMLTable() : Relational Views of XML

- The “COLUMNS” clause extends XMLTable, allowing the creation of in-line relational views of XML content
- Enables SQL operations on XML content
  - Views allow Non-XML aware tools access to XML content
- Collection hierarchy managed using chained XMLTable operations
  - Repeating elements passed down the chain as XMLType fragments

ORACLE

Copyright © 2017, Oracle and/or its affiliates. All rights reserved. |

XMLTable Columns Clause

```
SQL> select m.REFERENCE, l.LINENO, l.QUANTITY
  2     from PURCHASEORDER,
  3          XMLTable(
  4            '$PO/PurchaseOrder'     passing OBJECT_VALUE as "PO”
  5            COLUMNS
  6              REFERENCE                    VARCHAR2(32) PATH 'Reference',‘,
  7              LINEITEM_FRAGMENT            XMLTYPE PATH 'LineItems/LineItem’
  8         ) m,
  9         XMLTable(
 10            '$LI/LineItem'              passing m.LINEITEM_FRAGMENT as "LI”
 11           COLUMNS
 12             LINENO                      NUMBER(4) PATH '@ItemNumber‘,
 13             UPC                         NUMBER(14) PATH 'Part/text()‘,
 14             QUANTITY                    NUMBER(5) PATH 'Quantity‘
 15         )
 16 where l.UPC =  '24543000457’;
```

RERERENCE	LINENO	QUANTITY
AKHOO-20100418162507692PDT	2	2
PVARGAS-20101114171322653PST	1	7
JTAYLOR-20100518182653281PDT	5	4

Copyright © 2017, Oracle and/or its affiliates. All rights reserved. |

XQuery-Update support

Standards-compliant update of XML content

- http://www.w3.org/TR/xquery-update-10/
- Declaratively modify and transform XML content
  - Replace, Delete, Insert, Rename
- Database supports the copy / modify form of XQuery-Update
- Operations are re-written into partial-updates where possible
- XQuery update support is also enabled starting with patchset 11.2.0.3.0

ORACLE

Copyright © 2017, Oracle and/or its affiliates. All rights reserved. |

XQuery-Update support

```
update TABLE
    set XML_COLUMN =  xmlQUERY(
                                'copy $NEWXML := $XML modify (
                                  let $TARGET := $NEWXML/rootElement/targetElement
                                   return replace node $TARGET with $NEWCONTENT
                                )
                                return $NEWXML'
                                passing XML_COLUMN as "XML",  V_NEW_CONTENT as " NEWCONTENT "
                                returning content
                            )
where …
```

- Combine an XMLQuery operator containing an XQuery-Update expression with a SQL Update statement
  - The XQuery-Update supplies the new value for the XMLType

Copyright © 2017, Oracle and/or its affiliates. All rights reserved. |

XQuery-Update support

```
copy $NEWPO := $PO modify (
                        let $ROOT := $NEWPO/PurchaseOrder
                        let $SHIPINST := $ROOT/ShippingInstructions
                        return (
                            replace value of node $ROOT/User with $NEWUSER,
                            replace node $ROOT/Requestor with element Requestor {$NEWNAME},
                            delete node $SHIPINST/address,
                            insert node element street {$STREET} as last into $SHIPINST,
                            insert node element city {$CITY} as last into $SHIPINST,
                            insert node element zipcode {$ZIPCODE} as last into $SHIPINST
                        )
                    )
                    return $NEWPO
```

- Select target nodes using For or Let clauses
- Specify modifications in the return clause

Copyright © 2017, Oracle and/or its affiliates. All rights reserved. |

XQuery-Update usage

- Updating XML stored in the Oracle Database
- XML Data Masking
  - Use an XMLType view based on XQuery-Update to remove or mask content from an XML document
- Updating xml is supported using Oracle specific operators in older releases of the database
  - UpdateXML(), DeleteXML(), insertChildXML() etc
  - Legacy Oracle specific operators are deprecated in Oracle Database 12c

Copyright © 2017, Oracle and/or its affiliates. All rights reserved. |

XQuery-FullText support

Standards compliant Full-Text search of XML content

- XQFT http://www.w3.org/TR/xpath-full-text-10/
- Declarative full text search in XML content
  - contains text, stemming, windowing
  - Logical full text operators : ftor, ftand, ftMildNot
- The XQuery contains() operator is limited to ‘string within string’ searches
- Available starting with Oracle Database 12.1.0.1.0

Copyright © 2017, Oracle and/or its affiliates. All rights reserved. |

XQuery-FullText support

```
select PURCHASEORDER_XML
  from PURCHASEORDERS
where XMLExists(
          '$P/PurchaseOrder/ShippingInstructions/Address/street[. contains text {$PHRASE}]‘
          passing PURCHASEORDER_XML as "P“, 'sport' as "PHRASE“
);
```

- Use with XMLExists() operator to perform Full-Text filtering
- Use with XMLQuery() and XMLTable() to perform Full Text based fragement extraction
- Support for Document, Fragment, Element and Attribute level operations
- Document and Fragment level operations target all descendant nodes

Copyright © 2017, Oracle and/or its affiliates. All rights reserved. |

XQuery-FullText support

Common operations

```
$PO/PurchaseOrder/ShippingInstructions/Address/street[. contains text {$WORD} using stemming]
```

- Use Stemming to search for different forms a word

```
'$PO/PurchaseOrder/ShippingInstructions/Address[. contains text {$WORD1} ftand {$WORD2}
```

- Use FT logical operators to search for multiple words

```
$P/PurchaseOrder/ShippingInstructions/Address[. contains text {$WORD1} ftand {$WORD2} window 6 words]
```

- Use Windowing to specify the distance between words

Copyright © 2017, Oracle and/or its affiliates. All rights reserved. |

XQuery-FullText Indexing

- XQuery-FullText operations are optimized by XML aware Full-Text (Context) index
- Based on existing Oracle Text (Context) Technology
- Proven Full-Text search capabilities
- Fully Namespace Aware
- Use XQuery Full-Text rather than SQL's contains() operator when working with XML content
- The oracle xpath extension ora:contains() is depricated starting with Oracle Database 12c

Copyright © 2017, Oracle and/or its affiliates. All rights reserved. |

XQuery-FullText Index Creation

```
BEGIN
  CTX_DDL.CREATE_SECTION_GROUP('XQFT','PATH_SECTION_GROUP');
  CTX_DDL.SET_SEC_GRP_ATTR('XQFT','XML_ENABLE','T');
  CTX_DDL.create_preference('STORAGE_PREFS', 'BASIC_STORAGE');
  CTX_DDL.set_attribute( 'STORAGE_PREFS',
             'D_TABLE_CLAUSE', 'LOB(DOC) STORE AS SECUREFILE (COMPRESS MEDIUM CACHE)‘ );
  CTX_DDL.set_attribute( 'STORAGE_PREFS',
              'I_TABLE_CLAUSE', 'LOB(TOKEN_INFO) STORE AS SECUREFILE (NOCOMPRESS CACHE)');
END;
/

CREATE INDEX PURCHASEORDER_XQFT_IDX
   ON PURCHASEORDER(OBJECT_VALUE)
     INDEXTYPE IS CTXSYS.CONTEXT
     PARAMETERS(
       'storage STORAGE_PREFS|
       section group XQFT’
     )
/
```

Copyright © 2017, Oracle and/or its affiliates. All rights reserved. |

Other SQL/XML Operators

- XMLCast()
  - Used with XMLQuery() operator
  - Converts XML scalars into SQL scalars
- XMLTransfom()
  - XSL based transformation
- XMLNamespaces()
  - Namespace management
- SchemaValidate()
  - XMLType method for validating document against an XML Schema

Copyright © 2017, Oracle and/or its affiliates. All rights reserved. |

XML Storage and Indexing

Copyright © 2017, Oracle and/or its affiliates. All rights reserved. |

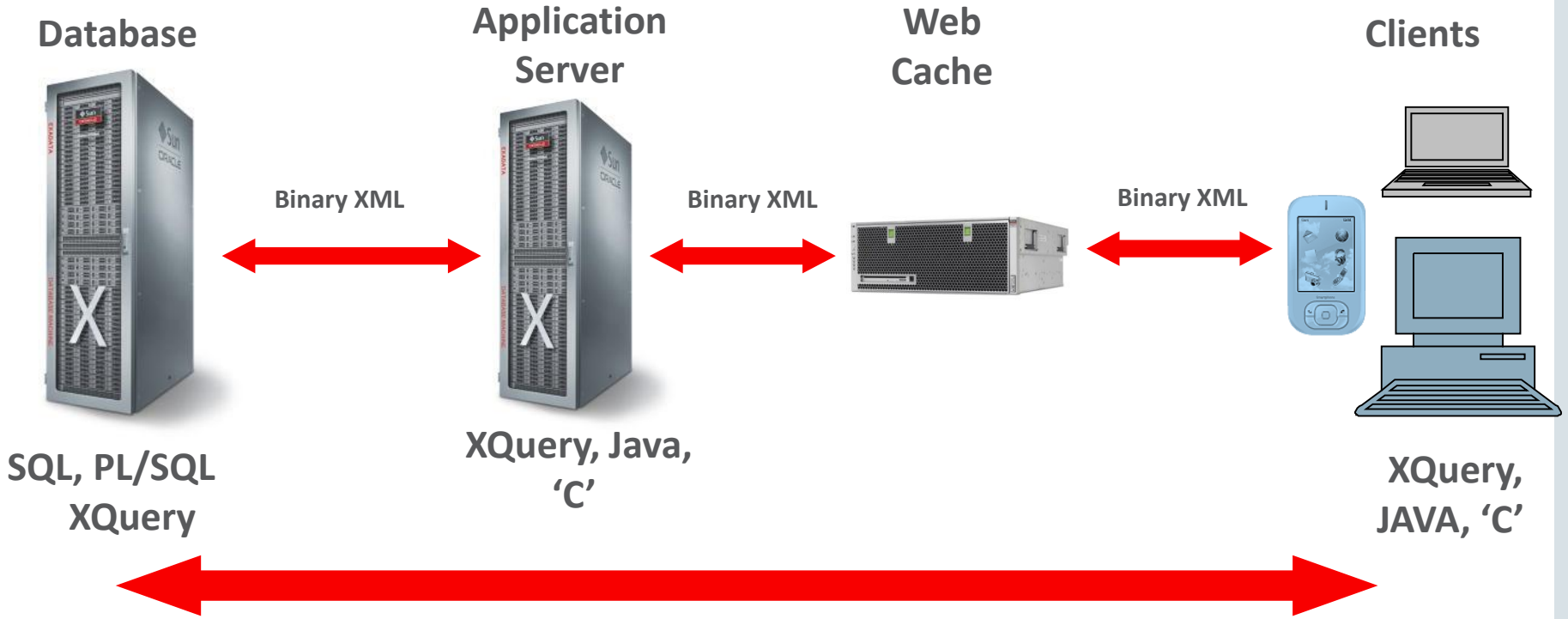
Binary Persistence

```
SQL> create table PURCHASEORDER of XMLTYPE
  2> XMLTYPE store as SECUREFILE BINARY XML;
```

- Stores post-parse representation of XML on disc
  - Reduced storage requirements
  - Tags are tokenized, content stored in native representation
- Optimized for streaming, indexing and fragment extraction.
- Single representation used on disc, in-memory and on-wire
  - No parsing / serialization overhead once XML is ingested
- Partial update
- Schema-less and XML Schema aware versions

Copyright © 2017, Oracle and/or its affiliates. All rights reserved. |

Oracle Binary XML



Copyright © 2017, Oracle and/or its affiliates. All rights reserved. |

Client Side Encoding

```
XMLType xml = new XMLType(connection,is);
xml.setPicklePreference(XMLType.XMLTYPE_PICKLE_AS_BINXML);
xml.setMetadataConn(connection);
statement.setObject(1,xml);
boolean result = statement.execute();
```

- Enable client side encoding by executing the methods setPicklePreference() and setMetadataConn() on the XMLType object before performing the Insert
  - Specificy XMLType.XMLTYPE_PICKLE_AS_BINXML when invoking setPicklePreference() to force client side encoding.
- Client side encoding is only supported with XMLType.
  - Currently it is not available when using the JDBC 4.0 SQLXML class

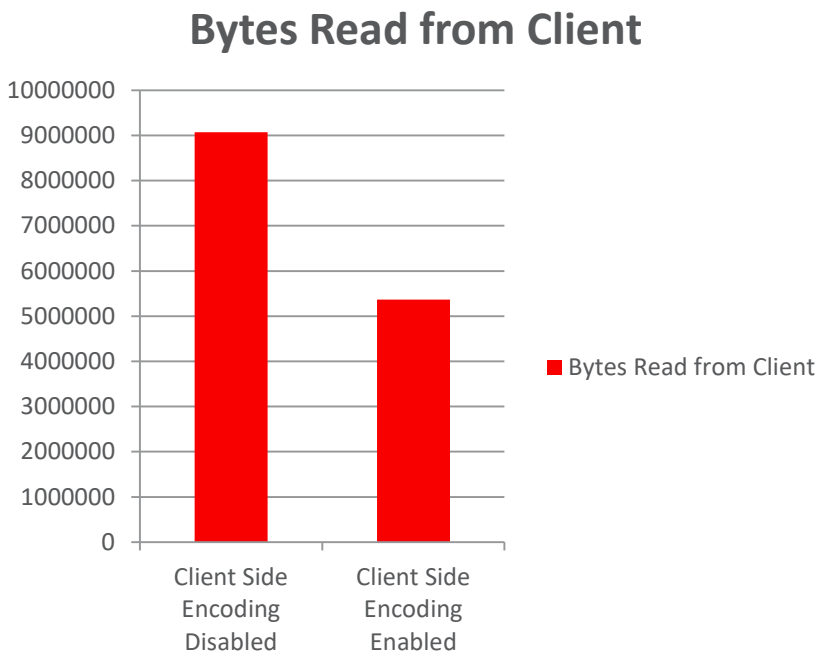
Copyright © 2017, Oracle and/or its affiliates. All rights reserved. |

Client Side Encoding

- Allows the overhead of parsing and encoding the XML to be moved from the database to the client
  - Reduction in overall Network Traffic
  - Reduction in Database CPU usage
  - Increase in Client CPU usage (no free lunch)
- Most effective when many similar documents will be inserted using a single database connection
- Recommended when constructing XMLType from InputStream.
- Avoid when constructing XMLType directly from XMLDocument

Copyright © 2017, Oracle and/or its affiliates. All rights reserved. |

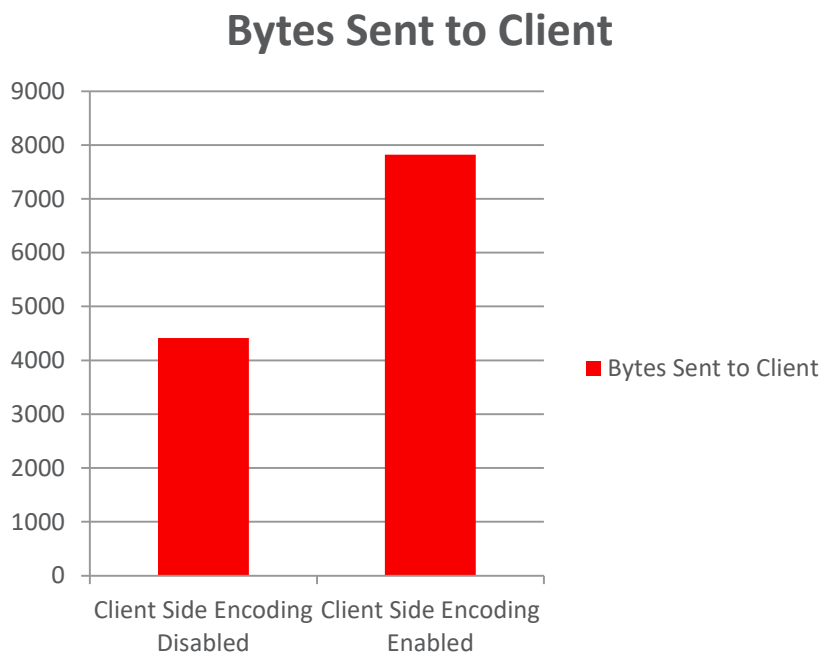
Client Side Encoding : Bytes sent to Server



- 50% reduction in bytes sent to server
- Complete Works of Shakespeare
- 36 Files, 9,067,913bytes
- High content to Tag ratio

Copyright © 2017, Oracle and/or its affiliates. All rights reserved. |

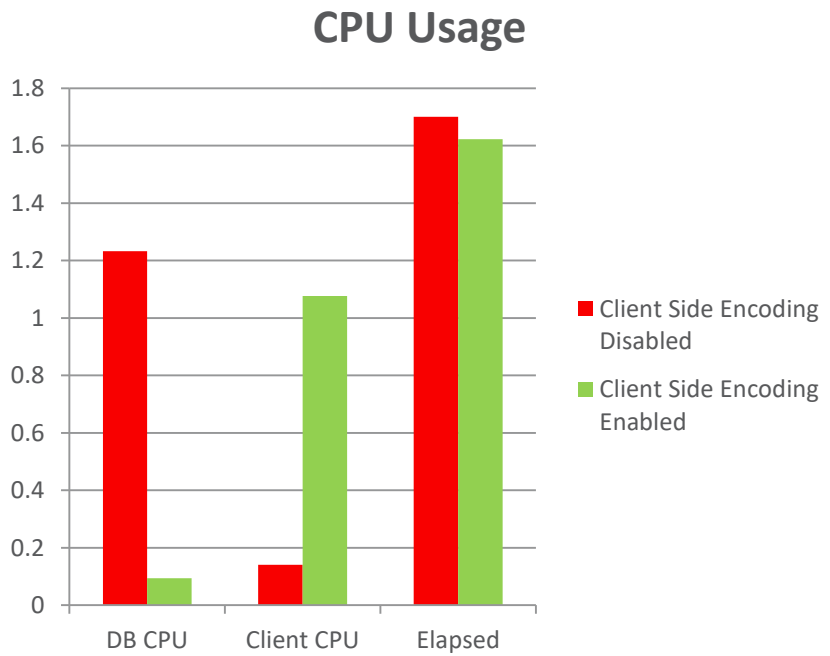
Client Side Encoding : Bytes sent to Server



- 2x bytes read from server
- Client needs token table to encode XML
- Less than .001% of content

Copyright © 2017, Oracle and/or its affiliates. All rights reserved. |

Client Side Encoding : CPU Usage



- 85% reduction in DB CPU
- 8x increase in Client CPU
- 5% reduction in elapsed time
- Significant improvement in scalability

Copyright © 2017, Oracle and/or its affiliates. All rights reserved. |

XML Index : Unstructured Index

```
SQL> create index PURCHASEORDER_XML_IDX
  2     on PURCHASEORDER (OBJECT_VALUE)
  3       indextype is XDB.XMLINDEX;
```

- Requires no knowledge of the structure of the XML being indexed or the search criteria
- All elements and attributes in the XML are indexed
  - Name / Value pair model
- Optimizes searching and fragment extraction
- Accelerates path and path-with-predicate searching
- Supports type-aware searches
- Synchronous and Asynchronous indexing modes

Copyright © 2017, Oracle and/or its affiliates. All rights reserved. |

Path Subsetting

```
SQL> create index PURCHASEORDER_XML_IDX
 2                  on PURCHASEORDER (OBJECT_VALUE)
 3                      indextype is XDB.XMLINDEX
 4                      parameters (
 5                        'paths (
 6                        include (
 7                           /PurchaseOrder/Reference
 8                           /PurchaseOrder/LineItems//* ))‘
 9                      );
```

- Indexing all nodes can be expensive
  - DML Performance & Space Usage
- Path sub-setting allows control over which nodes are indexed
- Trade off retrieval performance, DML performance and space usage

Copyright © 2017, Oracle and/or its affiliates. All rights reserved. |

Structured XML Index

```
SQL> create index PURCHASEORDER_XML_IDX
 2                  on PURCHASEORDER (OBJECT_VALUE)  indextype is XDB.XMLINDEX
 4                      parameters ('PARAM PO_SXI_PARAMETERS');
```

- Indexes “Islands of Structure”
  - Requires some knowledge of the XML being indexed and the kind of queries that will be performed
- Specific leaf-level nodes projected into relational tables
  - Table for each island, leaf node values stored as columns
- Data type aware
- Based on XMLTable syntax()
- Optimizes all SQL/XML operators
  - XMLQuery(), XMLTable() and XMLExists()

Copyright © 2017, Oracle and/or its affiliates. All rights reserved. |

XML Table Index DDL

```
SQL> call DBMS_XMLINDEX.registerParameter(
 2       'PO_SXI_PARAMETERS‘,
 3       'GROUP   PO_LINEITEM
 4            xmlTable PO_INDEX_MASTER
 5              ''/PurchaseOrder''
 6              COLUMNS
 7                  REFERENCE              varchar2(30)              PATH ''Reference'',
 8                  LINEITEM               xmlType                   PATH ''LineItems/LineItem''
 9            VIRTUAL XMLTABLE             PO_INDEX_LINEITEM
10              ''/LineItem''
11              PASSING lineitem
12              COLUMNS
13                  ITEMNO                 number(38)                PATH ''@ItemNumber'',
14                  UPC                    number(14)                PATH “Part/text()'',
15                  DESCRIPTION            varchar2(256)             PATH ''Part/@Description''
16     ');
```

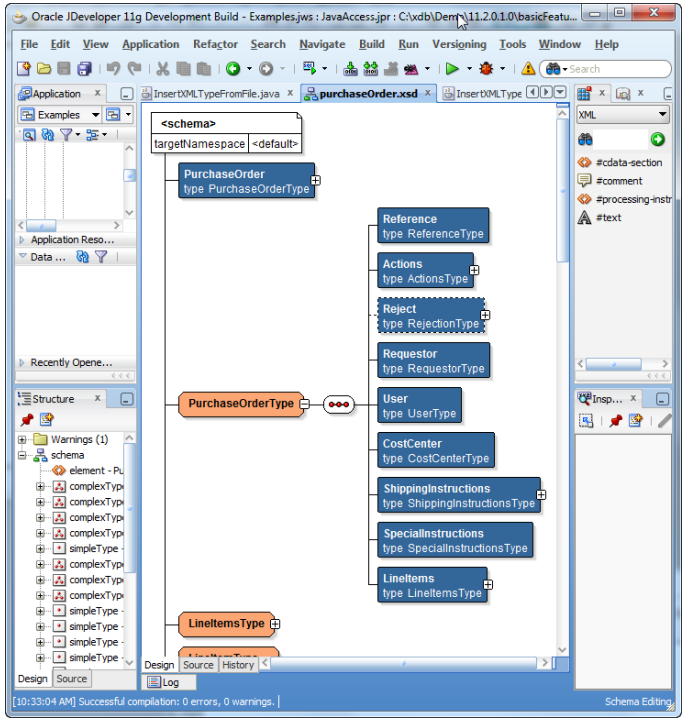
Copyright © 2017, Oracle and/or its affiliates. All rights reserved. |

XML Schema

ORACLE

Copyright © 2017, Oracle and/or its affiliates. All rights reserved. |

XMLSchema



- WC3 Standard for defining the structure and content of an XML document
  - An XML Schema is an XML document
- Used for validation purposes
  - Parsers like Oracle XDK, XERCES or Microsoft's MSXML
  - XML Editors like XMetal, Oxygene or Microsoft Word 2K7
- Created using tools like Altova's XML Spy or Oracle's JDeveloper

Copyright © 2017, Oracle and/or its affiliates. All rights reserved. |

XML Schema and Binary XML

```
DBMS_XMLSCHEMA.registerSchema  (
   SCHEMAURL              => 'http://www.example.com/xsd/purchaseOrder.xsd',
   SCHEMADOC        => xmlType(bfilename('XMLDIR','po.xsd'), nls_charset_id('AL32UTF8')),
   GENTYPES           => FALSE,
   GENTABLES              => FALSE,
   OPTIONS                => DBMS_XMLSCHEMA.REGISTER_BINARYXML )
```

- Increased storage efficiency for Binary XML
  - Simple types mapped to native formats
  - Pre-generated token tables
- Improves streaming XPath and XML Index operations
  - Leverages cardinality and location information
- Schema validation part of Binary XML encoding process

Copyright © 2017, Oracle and/or its affiliates. All rights reserved. |

Object-Relational Storage

```
DBMS_XMLSCHEMA.registerSchema  (
   SCHEMAURL              => 'http://www.example.com/xsd/purchaseOrder.xsd',
   SCHEMADOC        => xmlType(bfilename('XMLDIR','po.xsd'), nls_charset_id('AL32UTF8')),
   GENTYPES           => TRUE,
   GENTABLES              => TRUE )
```

- XML Schema defines an XML object Model,
- XML Schema compilation
  - SQL object model generated from the XML object model
  - Object-relational tables created to provide efficient storage for SQL objects.
- Object Relational storage enables
  - Lossless, bi-directional mapping between XML object model and SQL object model
  - XQuery execution via re-write into SQL operations on the underlying tables

ORACLE

Copyright © 2017, Oracle and/or its affiliates. All rights reserved. |

Object Relational Persistence

- Suitable for highly structured XML use-cases
  - XML collection hierarchy persisted as master/ details relationships using nested tables
  - Simple recursive structures handled automatically using out-of-line tables
- Near-relational performance for
  - Leaf level access and update
  - Collection manipulation (insert,delete)
- Indexing via B-Tree and Bitmap indexes
- Significant reductions in storage Vs serialized form
- Some overhead incurred for document-level storage and retrieval operations

ORACLE

Copyright © 2017, Oracle and/or its affiliates. All rights reserved. |

XML DB Repository

ORACLE

Copyright © 2017, Oracle and/or its affiliates. All rights reserved. |

Oracle XML DB Repository

- Organize and access content as files in folders rather than rows in tables
  - Manages XML and non-XML content
  - Native support for HTTP, FTP and WebDAV protocols
  - Content accessible using standard desktop Tools
- Enables document-centric development paradigm
  - Path based access to content
  - Queries based on location
- Hierarchical Index
  - Patented, high performance folder-traversal operations and queries

Copyright © 2017, Oracle and/or its affiliates. All rights reserved. |

Oracle DB repository Features

- Document Level Security
  - Highly granular access control
  - Database and application defined principles with custom authentication schemes
- Standards compliant
  - File / Foldering model : IETF WebDav standard
  - Security : DavACL
  - Protocols : HTTP, HTTPS, WebDAV
- Application Development
  - JCR Provider
  - SQL API and Views
  - Database Native Web Services

Copyright © 2017, Oracle and/or its affiliates. All rights reserved. |

Oracle DB repository XML features

- Schema based XML content automatically stored in user tables specified during schema registration
- Enables URL centric standards like XLink and Xinclude
- Special handling for office productivity documents
- Support for virtual documents
  - Based on XMLType views
  - Content generated when document is opened

Copyright © 2017, Oracle and/or its affiliates. All rights reserved. |

Content Management Features

- Access control
  - Grant / Revoke permissions on a document by document basis
- Versioning
  - Simple linear versioning model with Check-In and Check Out
- Comprehensive event model
  - Associate code with operations on files and folders
- Standard and user defined Metadata
  - Manage metadata independently from content
  - Java content management system API

Copyright © 2017, Oracle and/or its affiliates. All rights reserved. |

Database Native Web Services

Copyright © 2017, Oracle and/or its affiliates. All rights reserved. |

Database-native Web Services

- ‘Zero-Development’, ‘Zero-Deployment’ solution for publishing PL/SQL packages.
  - Any package method, function or procedure can be accessed as a SOAP end-point
- Leverages the Oracle XML DB HTTP Server
  - No additional infrastructure required
- Automatic generation of WSDL
  - URL to Package, Function or Procedure mapping scheme
- Uses XML DB infrastructure for processing request and generating response
- Includes ‘SQL Query’ and ‘XQuery’ Services

ORACLE

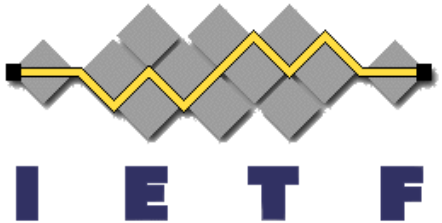
Copyright © 2017, Oracle and/or its affiliates. All rights reserved. |

Summary

Copyright © 2017, Oracle and/or its affiliates. All rights reserved. |

XML DB Summary

- **XMLType**
  - XML storage and indexing
- **XQuery, XML Schema, XSLT**
  - XML centric development
- **SQL/XML**
  - XML publishing
- **XMLDB repository**
  - XML specific content management
- **Standards** compliant
  - Strict adherence and conformance



Copyright © 2017, Oracle and/or its affiliates. All rights reserved. |

Integrated Cloud

Applications & Platform Services

Copyright © 2017, Oracle and/or its affiliates. All rights reserved. |

Q&A

Copyright © 2017, Oracle and/or its affiliates. All rights reserved. |

ORACLE®