ORACLE

Oracle XML DB in Oracle Database 19c and 23c

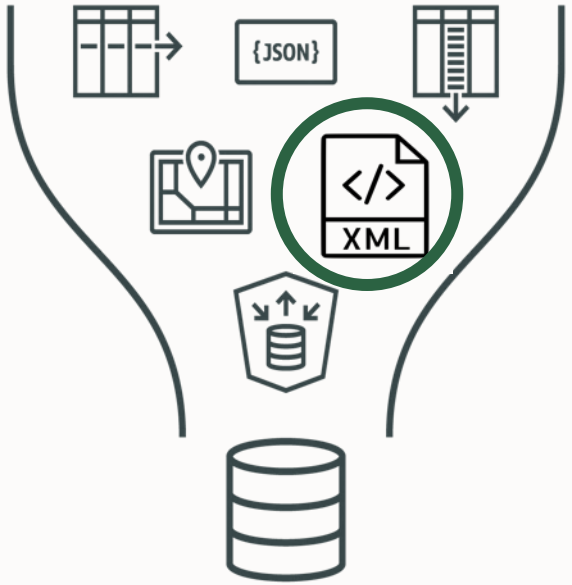
Technical Overview

Agenda

- **Introduction to XML DB**
- XMLTYPE
- SQL/XML
- XML Indexing
- XML Generation
- XML Schema Validation
- Demo and Summary
- References

Oracle Converged Database

Converged Database Architecture



for **any** data type or workload

XML data model for OLTP

- Fast document-centric CRUD operations

XML data model for Analytics

- Document-centric query, search, analytics and data integration

Multi-Model interoperability

- Support declarative multi-model transformation via SQL
- Support bi-directional transformations between hierarchical data and relational data

Agenda

- Introduction to XML DB
- **XMLTYPE**
- SQL/XML
- XML Indexing
- XML Generation
- XML Schema Validation
- Demo and Summary
- References

Sample XML Data

```
<PurchaseOrder  xmlns:xsi=http://www.w3.org/2001/XMLSchema-instance. xsi:noNamespaceSchemaLocation="purchaseOrder.xsd">
  <Reference>SBELL-20231009123336O1PDT</Reference>
   <User>SBELL</User>
  <CostCenter>S30</CostCenter>
  <ShippingInstructions> <name>Sarah Bell</name> <address>400 Oracle Parkway, Redwood Shores CA 94065  USA</address> </ShippingInstructions>
  <SpecialInstructions>Air Mail</SpecialInstructions>
  <LineItems>
    <LineItem ItemNumber="1">
      <Description>The Song of the Cell</Description>
      <Part Id="91982117354" UnitPrice="32.5" Quantity="2"/>
    </LineItem>
    <LineItem ItemNumber="2">
      <Description>The Unbearable Lightness Of Being</Description>
      <Part Id="37429140222" UnitPrice="29.95" Quantity="2"/>
    </LineItem>
    <LineItem ItemNumber="3">
      <Description>Immune</Description>
      <Part Id="90593241312" UnitPrice="36.95" Quantity="4"/>
    </LineItem>
  </LineItems>
</PurchaseOrder>
```

XMLTYPE

Native in-database storage of your documents

```
CREATE TABLE purchaseorder (
   po_id          NUMBER PRIMARY KEY,
   po_detail      XMLTYPE ) ;
```

Native SQL data type, makes database XML aware

- Use as Column, Variable, Argument or Return Value
- Optimized for streaming, indexing and fragment extraction

PL/SQL and Java constructors for creating an XMLTYPE instance

- VARCHAR2, CLOB, BLOB, or BFILE instance

Load table using SQL, JDBC, OCI, PLSQL and SQL Loader

XMLTYPE

Native in-database storage of your documents

23c

Binary XML in 19c

```
CREATE TABLE purchaseorder (
   po_id          NUMBER PRIMARY KEY,
   po_detail      XMLTYPE )
XMLTYPE COLUMN po_detail
STORE AS BINARY XML ;
```

Binary representation of XML

- No self-containment
- Default storage for compatible < 23.0

Transportable Binary XML in 23c

```
CREATE TABLE purchaseorder (
   po_id          NUMBER PRIMARY KEY,
   po_detail      XMLTYPE )
XMLTYPE COLUMN po_detail
STORE AS TRANSPORTABLE BINARY XML ;
```

Transportable Binary XML introduced in 23c

- Self-contained binary representation of XML
- Increased scalability and performance
- Default storage for compatible >= 23.0

Recommended future storage model

XMLTYPE

Migration from 19c to 23c

23c

Binary XML in 19c

```
CREATE TABLE purchaseorder (
   po_id          NUMBER PRIMARY KEY,
   po_detail      XMLTYPE )
XMLTYPE COLUMN po_detail
STORE AS BINARY XML ;
```

Transportable Binary XML in 23c

```
CREATE TABLE purchaseorder (
   po_id          NUMBER PRIMARY KEY,
   po_detail      XMLTYPE )
XMLTYPE COLUMN po_detail
STORE AS TRANSPORTABLE BINARY XML ;
```

Migrate to Transportable Binary XML in Oracle Database 23c

- Online Redefinition
- CTAS
- Datapump import with TRANSFORM=XMLTYPE_STORAGE_CLAUSE:' "TRANSPORTABLE BINARY XML " '

Agenda

- Introduction to XML DB
- XMLTYPE
- **SQL/XML**
- XML Indexing
- XML Generation
- XML Schema Validation
- Demo and Summary
- References

The Power of SQL meets XML Documents

SQL/XML, XPath and XQuery

SQL/XML is a general interface between the SQL and XQuery languages

- SQL/XML, part of ANSI SQL standard

XQuery is the W3C language for querying and updating XML data

- Includes XQuery Update and XQuery Full Text

XPath is a W3C Recommendation for navigating XML documents, a subset of the XQuery language

- XPath models an XML document as a tree of nodes, common XPath constructs are /, //, *, [ ]
- XPath and XQuery support a set of built-in functions such as substring, round, and not

SQL/XML Functions

XQuery functions

- XMLExists() : Filtering
- XMLQuery() : Fragment Extraction
- XMLCast() : Conversion to SQL type system
- XMLTable() : Projection

Other functions

- XMLSerialize(): Serializing XML data into a string or LOB
- XMLParse(): parsing XML Data into XMLTYPE instance
- XMLTransfom(): XSL based transformation
- XMLNamespaces(): Namespace management

XMLEXISTS() XQuery Predicates

```
SELECT p.po_detail  XML
   FROM purchaseorder p
      WHERE XMLEXISTS (
            '$PO/PurchaseOrder[Reference="SBELL-20231009123336O1PDT"]'
             PASSING p.po_detail as "PO"
                );
```

Used in SQL WHERE clause to filter rows based on an XQuery expression

- Evaluate whether or not a given document contains a node that matches an XPath expression

XMLQUERY() Fragment access

```
SELECT XMLQUERY (
                  '$PO/PurchaseOrder/ShippingInstructions'
                   PASSING p.po_detail as "PO"
                   returning content) XML
FROM purchaseorder p
WHERE XMLEXISTS (
                   '$PO/PurchaseOrder[Reference=$REF]'
                    PASSING p.po_detail as "PO",  'SBELL-20231009123336O1PDT' as "REF");
```

Query XML data, extract a fragment from each document

- RETURNING CONTENT indicates the result from the XQuery evaluation is either an XML document or a document fragment (return type is XMLTYPE)
- Bind variables are supplied via the “PASSING” clause

XMLCAST(XMLQUERY())

```
SELECT XMLCAST(XMLQUERY (
                          '/PurchaseOrder/Reference'
                           PASSING p.po_detail
                           returning content) AS VARCHAR2(30)) "Reference"
   FROM purchaseorder p
      WHERE XMLEXISTS (
                        '$PO/PurchaseOrder/LineItems/LineItem[1]/Part[@Id=$ID]'
                         PASSING p.po_detail as "PO", '91982117354' as "ID");
```

Cast the scalar value of an XML fragment extracted from the document to a SQL data type

XMLQUERY() XQuery-Update support

```
UPDATE table_name
         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 …
```

Standards-compliant update of XML content

- http://www.w3.org/TR/xquery-update-10/
- Combine XMLQUERY operator containing an XQuery-Update expression with SQL Update statement
- The XQuery-Update supplies the new value for the XMLTYPE

XMLTABLE() Generate Relational Views of XML

Map the result of an XQuery evaluation into relational rows and columns of a virtual table

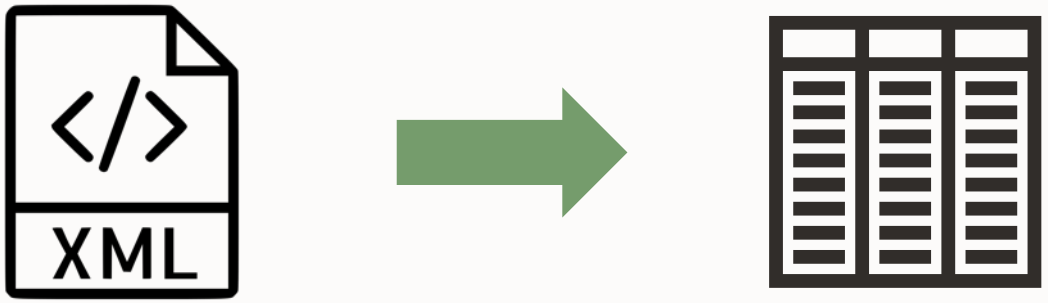
- Used in FROM clause
- The “COLUMNS” clause extends XMLTABLE, allowing creation of in-line relational views of XML content

Enable 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



XMLTABLE() Columns Clause

```
SELECT p.po_id, m.COLUMN_VALUE  description
       FROM purchaseorder p,
                      XMLTABLE (
                          '/PurchaseOrder/Reference' PASSING p.po_detail ) m;

SELECT p.po_id, m.reference, m.userid
       FROM purchaseorder p,
                      XMLTABLE (
                          '/PurchaseOrder'  PASSING p.po_detail
                           COLUMNS
                               reference       VARCHAR2(30)  PATH 'Reference',
                               userid          VARCHAR2(10)  PATH 'User',
                               ship_to_name    VARCHAR2(20)  PATH 'ShippingInstructions/name',
                               ship_to_address VARCHAR2(256) PATH 'ShippingInstructions/address') m;
```

Maps nodes in the XML document to columns in the XMLTABLE result

- One-to-one relationship between documents in the table with XMLTYPE column and the rows in the XMLTABLE result

Chained XMLTABLE() Clause

```
SELECT m.REFERENCE, i.LINENO, i.QUANTITY
       FROM purchaseorder p,
                     XMLTABLE ('$PO/PurchaseOrder'  PASSING p.po_detail as "PO"
                         COLUMNS
                             REFERENCE    VARCHAR2(30)  PATH 'Reference',
                             LINEITEM     XMLTYPE       PATH 'LineItems/LineItem'
                      ) m,
                      XMLTABLE ('$LI/LineItem'      PASSING m.LINEITEM as "LI"
                          COLUMNS
                             LINENO       NUMBER(4)     PATH '@ItemNumber',
                             UPC          NUMBER(12)    PATH 'Part/@Id',
                             QUANTITY     NUMBER(5)     PATH 'Part/@Quantity'
                      ) i
       WHERE i.UPC = '91982117354';
```

Apply XMLTABLE at each level when there is a one-to-many (1:N) relationship between documents in the table and rows in the XMLTABLE result

- Each PurchaseOrder element contains a LineItems element, which contains one or more LineItem elements that are mapped to XMLTYPE column passed to the second XMLTABLE

Agenda

- Introduction to XML DB
- XMLTYPE
- SQL/XML
- **XML Indexing**
- XML Generation
- XML Schema Validation
- Demo and Summary
- References

Structured XML Index

Indexes **fixed, structured part** of XML Content

- Requires some knowledge of the XML being indexed and the queries that will be performed
- Data type aware
- Based on XMLTABLE syntax

Specific leaf-level nodes projected into relational tables

- Table for each part, leaf node values stored as columns
- Very fast extraction, aggregations over leaf nodes

Optimizes the query with XMLTABLE expression

Structured XML Index DDL

```
CREATE INDEX po_index
      ON purchaseorder (po_detail) INDEXTYPE is XDB.XMLINDEX PARAMETERS (
            'XMLTABLE po_ptab
                ''/PurchaseOrder''
                 COLUMNS
                     reference  VARCHAR2(30)  PATH ''Reference'',
                      lineitem   XMLTYPE      PATH ''LineItems/LineItem'' VIRTUAL
             XMLTABLE li_tab
                ''/LineItem''  PASSING lineitem
                 COLUMNS
                      lineno    NUMBER(4)     PATH ''@ItemNumber'',
                      upc       NUMBER(12)    PATH ''Part/@Id'',
                       quantity  NUMBER(5)    PATH ''Part/@Quantity''
            ');
```

Index structured components

- Top level index table PO_PTAB has columns reference and lineitem
- Column lineitem is of type XMLTYPE which is virtual. It represents a collection, is passed to the second XMLTable construct to form the second-level relational index table, LI_TAB, which has columns lineno, upc, and quantity

Relational Views Over XML Data

```
CREATE OR REPLACE VIEW po_view AS
      SELECT p.po_id, m.reference, i.lineno, i.upc, i.quantity
           FROM purchaseorder p,
                  XMLTABLE('/PurchaseOrder'  PASSING p.po_detail
                     COLUMNS
                          reference      VARCHAR2(30)   PATH 'Reference',
                          lineitem       XMLTYPE        PATH 'LineItems/LineItem' ) m,
                   XMLTABLE (
                     '/LineItem'  PASSING m.lineitem
                     COLUMNS
                          lineno         NUMBER(4)      PATH '@ItemNumber',
                          upc            NUMBER(12)     PATH 'Part/@Id',
                          quantity       NUMBER(5)      PATH 'Part/@Quantity') i;
```

```
SELECT p.reference, p.lineno FROM po_view t, scott.order s WHERE t.reference =
s.reference;
```

Queries are optimized through Structured XML Index directly from the underneath index tables

Applications and tools can work on relational views directly without knowing the XML content

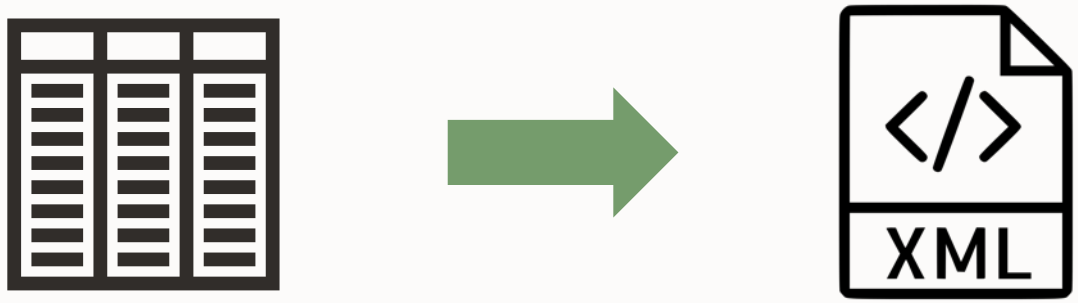
Agenda

- Introduction to XML DB
- XMLTYPE
- SQL/XML
- XML Indexing
- **XML Generation**
- XML Schema Validation
- Demo and Summary
- References

Generate XML from relational tables

Expose relational data as XML documents using SQL/XML

- XMLElement()
  - Generates an Element with simple or complex content
- XMLAttributes()
  - Adds attributes to an element
- XMLAgg()
  - Generates an XML Fragment
  - Aggregation operator used to process the results of a nested sub-query
- XMLForest, XMLConcat, XMLComment and more



XML Generation using SQL/XML

```
SELECT
  XMLElement ( "Department",
     XMLAttributes ( d.DEPTNO as "Id"),
     XMLElement ("Name", d.DNAME),
     XMLElement ("Employees", (
          SELECT XMLAgg (
                    XMLElement ("Employee",
                         XMLAttributes (e.EMPNO as "Id"),
                         XMLForest (
                              e.ENAME as "Name",
                              e.HIREDATE as "StartDate"
                                   )
                              )
                        )
          FROM emp e
          WHERE e.deptno = d.deptno
                              )
                  )
               ) xml
 FROM dept d;
```

XML

```
<Department Id="10">
 <Name>ACCOUNTING</Name>
 <Employees>
   <Employee Id="7782">
     <Name>CLARK</Name>
     <StartDate>1981-06-09</StartDate>
   </Employee>
   <Employee” Id=‘7839”>
     <Name>KING</Name>
     <StartDate>1981-11-17</StartDate>
   </Employee>
   <Employee Id=“7934”>
     <Name>MILLER</Name>
     <StartDate>1982-01-23</StartDate>
   </Employee>
 </Employees>
</Department>
```

Agenda

- Introduction to XML DB
- XMLTYPE
- SQL/XML
- XML Indexing
- XML Generation
- **XML Schema Validation**
- Demo and Summary
- References

XML Schema Validation

Validate XML documents using DBMS_XMLSCHEMA_UTIL

Introduced in 19c, available on-premises and in Autonomous Database

Validates an xml document (data type is XMLTYPE) against an xml schema (data type is XMLTYPE)

API definition

```
procedure validate(doc in XMLTYPE, sch in XMLTYPE)
function conforming(doc in XMLTYPE, sch in XMLTYPE) return number
```

- Procedure raises ORA-31154 if either the schema is not legal, or the document does not conform to the schema
- Function returns zero if the schema is legal and the document conforms to the schema; otherwise, it returns an error code.

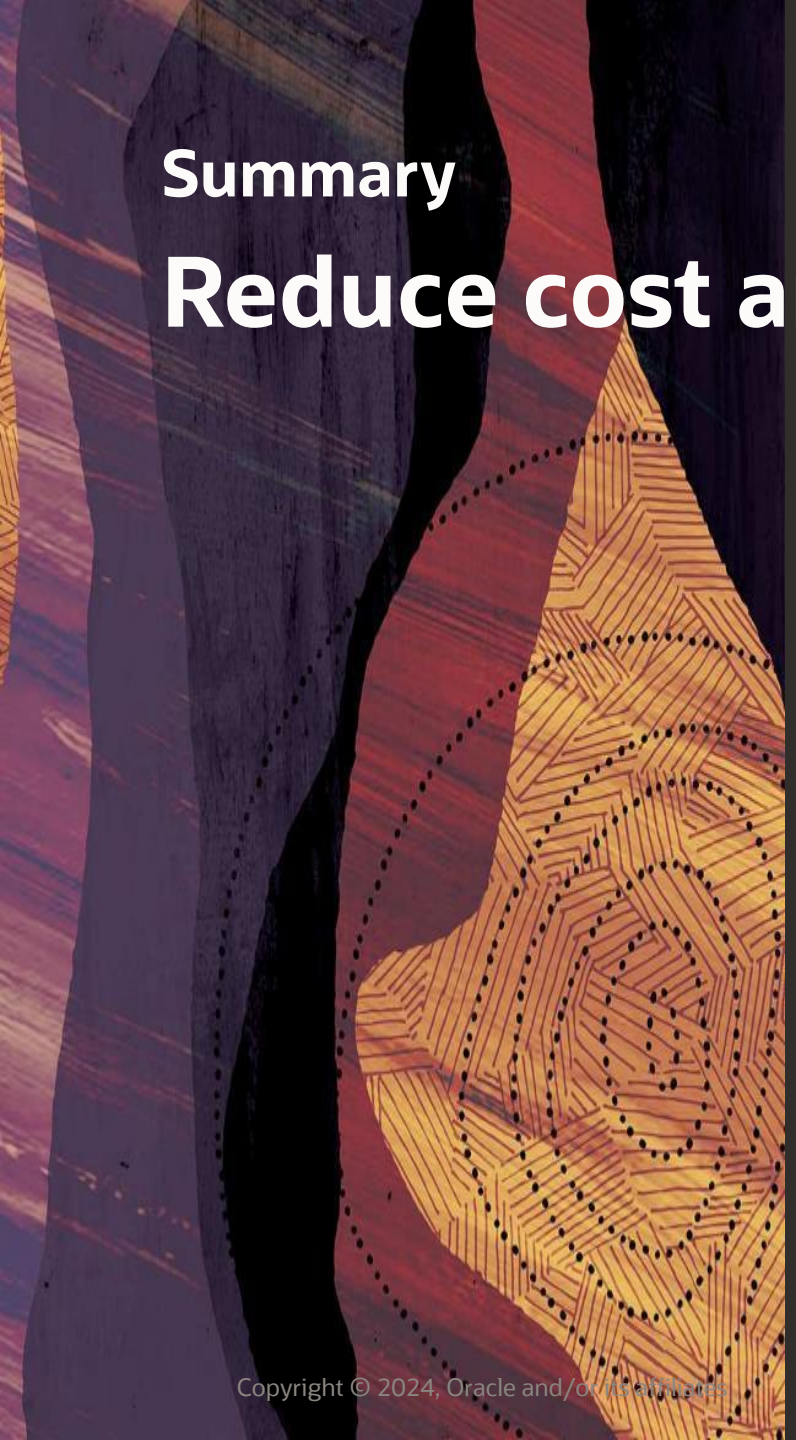
Note: DBMS_XMLSCHEMA.registerSchema() is not available on Oracle Autonomous Database

XML Schema Validation

```
declare
   sch      XMLTYPE;
   doc      XMLTYPE;
   status number;
begin
   sch := XMLTYPE('……');
   doc := XMLTYPE('……');
   DBMS_XMLSCHEMA_UTIL.VALIDATE(doc, sch);

   status := DBMS_XMLSCHEMA_UTIL.CONFORMING(doc, sch);

    if (status = 0) then

      -- load data into table

    end if;
end;
```

Applications can validate the document instance before loading into the table

Demo



Summary

Reduce cost and risk. Simplify your work.

1 Store, use, and manage relational data and XML documents in a single converged database. Unified management, security, consistency model

2 Flexible relational and XML access of your data, leveraging Oracle Database's core enterprise features for availability, security, scalability, and performance.

Where to get more information

Oracle.com

- Oracle XML DB - https://www.oracle.com/database/technologies/appdev/xmldb.html
- Oracle Autonomous Database - https://www.oracle.com/database/autonomous-database.html

Documentation

- XML DB Developer Guide - https://docs.oracle.com/en/database/oracle/oracle-database/23/adxdb/index.html

Livelab

- Oracle XML DB - https://apexapps.oracle.com/pls/apex/dbpm/r/livelabs/view-workshop?wid=3661

Thank you


ORACLE

Our mission is to help people see data in new ways, discover insights, unlock endless possibilities.