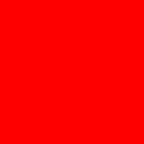


**ORACLE®**





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.

# Agenda

- Oracle XML Developer's Kit Overview
- XQuery Language Overview
- XQuery XQJ API Details
- XQJ Examples



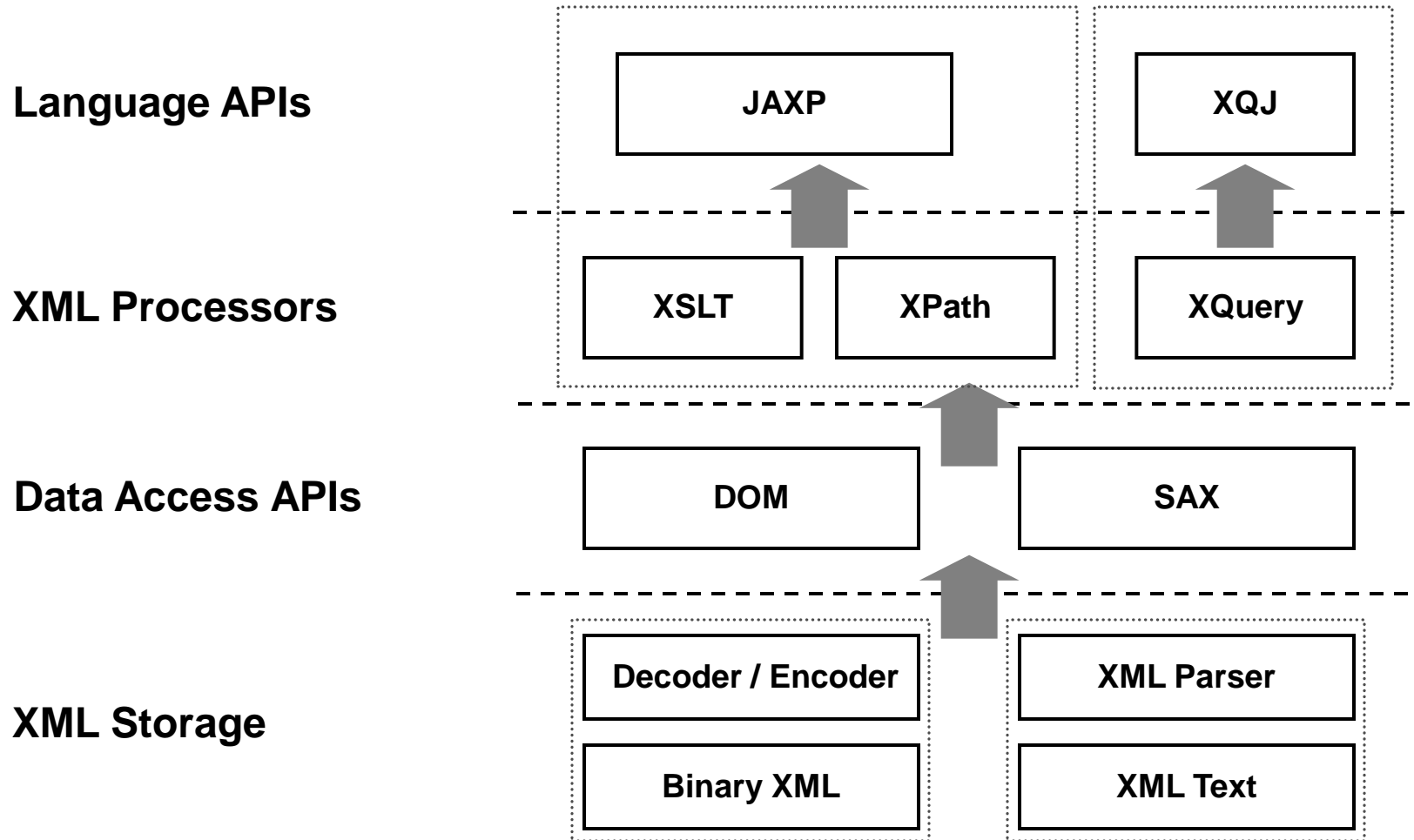
# Oracle XML Developer's Kit (XDK)

- Development Kit for XML processing
- Standalone / Middleware environments
  - XDK for Java
  - XDK for C/C++
- XML Data Model APIs
  - XML Parser, SAX, DOM
- XML Processing Languages
  - XPath, XSLT, XQuery
- Oracle XML DB Connectivity
  - XQuery XQJ API for accessing XML DB

# XDK Standards Support

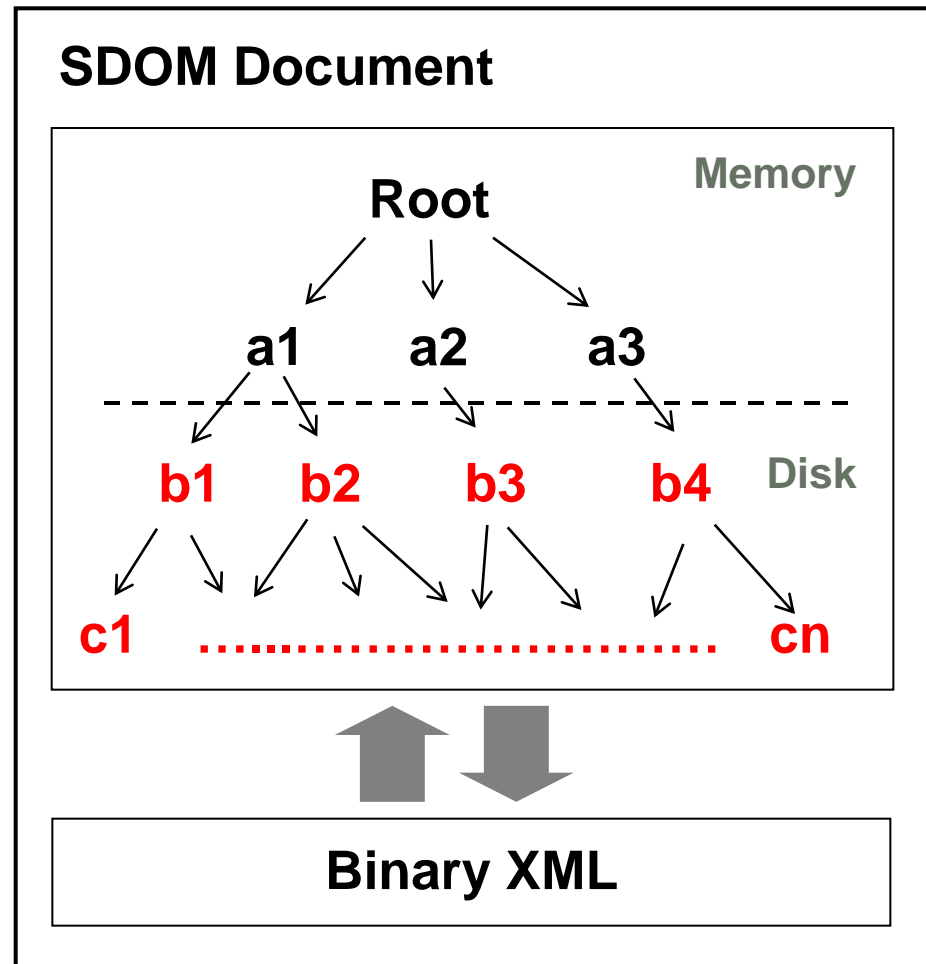
Standard	Version	Language
XML	1.0	Java, C/C++
XML Namespaces	1.0	Java, C/C++
DOM	1.0/2.0/3.0	Java, C/C++
SAX + Extensions	1.0/2.0	Java, C/C++
XSLT + XPath	1.0/2.0	Java, C/C++
XML Schema	1.0	Java, C/C++
JAXP	1.2	Java
JAXB	1.0	Java
JCR	1.0	Java
XQuery + XQJ API (JSR-225),	1.0	Java

# XDK Architecture Diagram



# Scalable XML Storage

- Scalable DOM
  - Backed by disk storage
  - Nodes lazily created
  - Unused nodes can be garbage collected
  - Less memory usage
  - GB-scale documents



# Scalable XML Processing

- Scalable DOM as input to all XML processors
  - Support for large XML documents
- Streaming XPath and XQuery processing
  - Incremental (lazy) evaluation
  - No in-memory XML materialization for streamable queries
  - Streaming input and output support in XQJ API
    - `java.io.InputStream`
    - `jaxax.xml.stream.XMLStreamReader` (StAX)
    - SAX events

# XQuery Overview

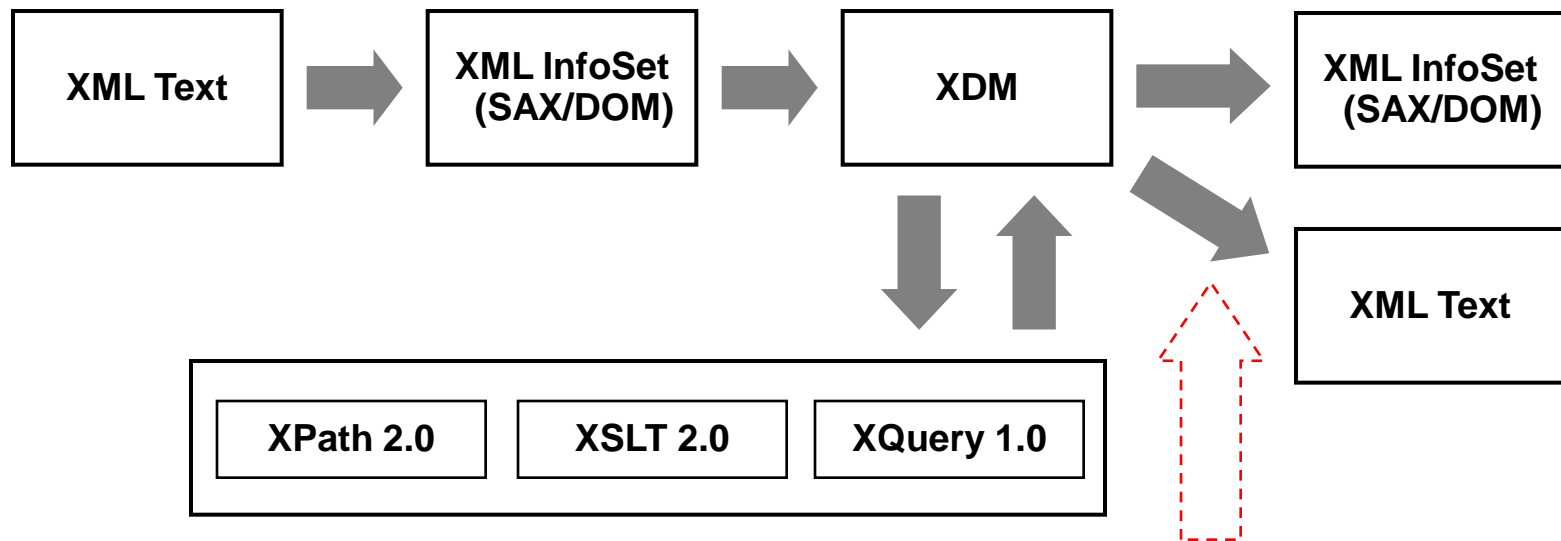
- XQuery – XML Query Language
- W3C Standard
  - W3C Recommendation 23 January 2007
- Declarative language for XML processing
- Concise and easy to learn
- XML construction, navigation, and sorting
- XML Schema validation
- Standard function library
- User-defined functions and modules
- External functions implemented in host language

# W3C XQuery Specifications

- XQuery 1.0
  - <http://www.w3.org/TR/xquery/>
- XQuery 1.0 and XPath 2.0 Data Model (XDM)
  - <http://www.w3.org/TR/xpath-datamodel/>
- XQuery 1.0 and XPath 2.0 Functions and Operators
  - <http://www.w3.org/TR/xpath-functions/>
- XQuery Update Facility 1.0
  - <http://www.w3.org/TR/xquery-update-10/>
- XQuery and XPath Full Text 1.0
  - <http://www.w3.org/TR/xpath-full-text-10/>
- XQuery Scripting Extension 1.0
  - <http://www.w3.org/TR/xquery-sx-10/>
- XSLT 2.0 and XQuery 1.0 Serialization
  - <http://www.w3.org/TR/xslt-xquery-serialization/>

# XQuery 1.0 and XPath 2.0 Data Model (XDM)

- Defines Input and Output for W3C XML languages
  - XPath 2.0, XSLT 2.0, XQuery 1.0
- XDM = Sequence of XML Nodes and Simple Values



- XSLT 2.0 and XQuery 1.0 Serialization (XDM to Text)

# XQuery Language Components

- XPath for XML navigation  
`doc("customer")/customers/customer`
- XML node construction  
`<customer><name>{ data($x/name) }</name></customer>`
- Iteration, filtering, joins and sorting (FLWOR expression)  
`for $c in doc("customer")//customer  
where $c/zip_code = 94065  
order by $c/last_name  
return <cust>{ $c/last_name, $c/address }</cust>`
- If-then-else  
`if ($c/max_order_amount > 1000)  
then <grp_1>{$c}</grp_1> else <grp_2>{$c}</grp_2>`
- Validation  
`import schema "http://www.example.org/customer";  
validate { doc("customer")//customer }`

# XQuery Update Facility (XQUF)

- W3C **Candidate Recommendation** 09 June 2009
- Extension of XQuery to perform updates on XML documents
- Insert/Delete/Replace/Rename XML nodes
- Example

```
for $c in doc("customer")//customer
where
    $c/max_order_amount > 1000
return
    replace value of node $c/discount
    with $c/discount * 2
```

# XQuery and XPath Full Text

- W3C [Candidate Recommendation](#) 28 January 2010
- Extension of XPath and XQuery for Full-Text Search in XML Documents
- Example

```
for $book score $score in doc("bib.xml")//book
  [title contains text "oracle" ftand "database"]
where $score > 0.5
order by $score descending
return
<result>
  <title>{ data($book/title) }</title>
  <score>{ $score }</score>
</result>
```

# XQuery Scripting Extension (XQ SX)

- W3C **Working Draft** 8 April 2010
- Original XQuery Language
  - side-effect free
  - order of evaluation is not defined
- XQuery Scripting Extension enhances XQuery
  - same syntax for language constructs (expressions)
  - defines exact evaluation order
  - allows for side-effects (updates) during evaluation
- XQ SX turns XQuery into the scripting language
  - Scripting language designed for XML processing

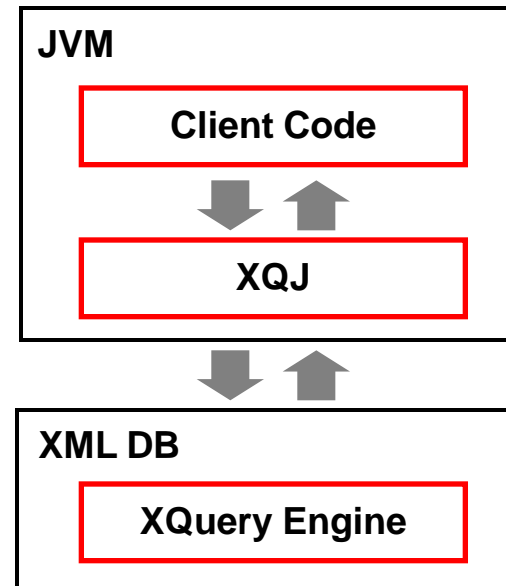
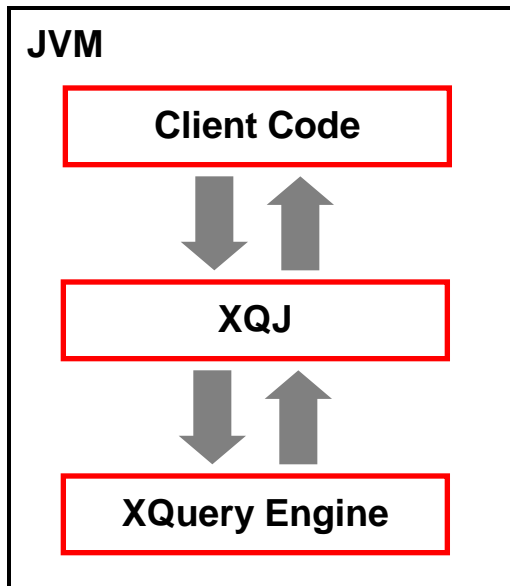
# XQJ Overview

- XQuery API for Java
- JSR 225 (<http://jcp.org/en/jsr/detail?id=225>)
  - Final release: 24 Jun, 2009
  - Reference implementation is available
- javax.xml.xquery package
- Conceptually very similar to JDBC
- Interoperability with existing XML APIs
  - SAX, DOM, StAX, both input and output
- Flexible runtime environments
  - XQuery engine collocated in the same JVM
  - Remote XQuery engine ( XML Database )

# XQJ Runtime Environments

XQJ for Java XQuery Engine

XQJ for XML Database



# XQJ – JDBC Comparison

JDBC	XQJ
<ul style="list-style-type: none"><li>• <code>javax.sql.DataSource</code></li></ul>	<ul style="list-style-type: none"><li>• <code>javax.xml.xquery.XQDataSource</code></li></ul>
<ul style="list-style-type: none"><li>• <code>java.sql.Connection</code></li></ul>	<ul style="list-style-type: none"><li>• <code>javax.xml.xquery.XQConnection</code></li></ul>
<ul style="list-style-type: none"><li>• <code>java.sql.Statement</code></li></ul>	<ul style="list-style-type: none"><li>• <code>javax.xml.xquery.XQExpression</code></li></ul>
<ul style="list-style-type: none"><li>• <code>java.sql.PreparedStatement</code></li></ul>	<ul style="list-style-type: none"><li>• <code>javax.xml.xquery.XQPreparedExpression</code></li></ul>
<ul style="list-style-type: none"><li>• <code>java.sql.ResultSet</code></li></ul>	<ul style="list-style-type: none"><li>• <code>javax.xml.xquery.XQResultSequence</code></li></ul>
<ul style="list-style-type: none"><li>• <code>java.sql.DatabaseMetaData</code></li></ul>	<ul style="list-style-type: none"><li>• <code>javax.xml.xquery.XQMetaData</code></li></ul>

# XQJ – Key Classes and Methods

- XQDataSource – Connection factory
  - getConnection() : XQConnection
  - getConnection(java.sql.Connection): XQConnection
- XQConnection – Expression factory
  - createExpression() : XQExpression
  - prepareExpression(<query\_content>) : XQPreparedExpression
- XQPreparedExpression – Variable binding and execution
  - bindDocument(), bindNode(), bindInt(), bindString(), etc
  - executeQuery() : XQResultSequence
- XQResultSequence – Result navigation
  - writeSequence(), writeSequenceToSAX(), getSequenceAsStream()
  - next(), first(), last()
  - writeItem(), getItemAsStream(), getNode(), writeItemToSAX()
  - getString(), getInt(), getBoolean(), etc

# Efficient and Scalable XQuery Evaluation with XQJ and XDK

- Input
  - Scalable DOM Node
  - Streamable data
    - `java.io.InputStream`, `javax.xml.stream.XMLStreamReader`
  - Deferred variable binding mode and lazy evaluation
    - Input consumed incrementally, only when needed
- Output
  - As Scalable DOM Node
  - Streaming of the result:
    - write to `java.io.OutputStream`,
    - get as `javax.xml.stream.XMLStreamReader`
    - convert to SAX events

# “Hello, World!” in XQJ/XQuery

```
import javax.xml.xquery.*
```

```
▶ XQDataSource dataSource = new oracle.xml.xquery.OXQDataSource();  
▶ XQConnection connection = dataSource.getConnection();  
▶ XQExpression expression = connection.createExpression();  
  
▶ XQResultSequence result = expression.executeQuery(  
    "<message> Hello, world! </message>  
");  
▶ result.writeSequence(System.out, null);  
  
result.close();
```

*Output:*

```
▶ <message> Hello, world </message>
```

# Prepared Expression, External Variables, DOM as Input/Output

```
import javax.xml.xquery.*

XQDataSource dataSource = new oracle.xml.xquery.OXQDataSource();
XQConnection connection = dataSource.getConnection();
▶ XQPreparedExpression preparedExpr = connection.prepareExpression(
    declare variable $doc external;
    for $c in $doc//customer
    where $c/LAST_NAME eq "Smith"
    return $c
);

▶ org.w3c.dom.Document scalableDOMDocument = ... // using JAXP's DocumentBuilder
▶ preparedExpr.bindNode(new QName("doc"), scalableDOMDocument);

▶ XQResultSequence result = preparedExpr.executeQuery();
while (result.next()) {
▶     org.w3c.dom.Node customerElement = result.getNode();
    // process `customerElement`
}
result.close();
```

# Streaming Evaluation using StAX API

```
import javax.xml.xquery.*

XQDataSource dataSource = new oracle.xml.xquery.OXQDataSource();
XQConnection connection = dataSource.getConnection();
XQPreparedExpression preparedExpr = connection.prepareExpression(
    declare variable $doc external;
    for $c in $doc//customer
    where $c/LAST_NAME eq "Smith"
    return $c
);

▶ javax.xml.stream.XMLStreamReader inReader = ... // using JAXP's XMLInputFactory
▶ preparedExpr.bindDocument(new QName("doc"), inReader, null);

▶ XQResultSequence result = preparedExpr.executeQuery();
while (result.next()) {
▶     XMLStreamReader customerElementReader = result.getItemAsStream();
    // process 'customerElementReader'
}
result.close();
```

# Using XQJ to access XML DB

```
import javax.xml.xquery.*

▶ java.sql.Connection sqlConnection = java.sql.DriverManager.getConnection(
    "jdbc:oracle:thin:@localhost:1251", "SCOTT", "TIGER"
);
XQDataSource xqDataSource = new oracle.xml.xquery.OXQDataSource();
▶ XQConnection xqConnection = xqDataSource.getConnection(sqlConnection);
▶ XQPreparedExpression xqPreparedExpr = xqConnection.prepareExpression(
    "declare variable $s as xs:int external;
    for $e in doc('/public/emp.xml')/emp/emp
    where $e/@salary > $s
    return $e"
);
▶ xqPreparedExpr.bindInt(new QName("s", 100000), null);
▶ XQResultSequence xqResult = xqPreparedExpr.executeQuery();
while (xqResult.next()) {
▶     org.w3c.dom.Node employeeElement = xqResult.getNode();
    // process 'employeeElement'
}
xqResult.close();
```

# Oracle XML DB DEMOgrounds Booths

- **Come by our DEMOgrounds booths to have one-on-one conversation with our team members**
  - Moscone West: W-41, W-44, and W-61

## Thursday Sessions

### **S317504: Waters Use Case and Structured XMLIndex**

Moscone South, Rm 200

10:30 AM – 11:30 AM

### **S317528: Working with Complex XML Schemas: Not as Hard as You Might Think**

Hotel Nikko Nikko Ballroom I

2:00 PM – 3:00 PM

### **S317657: XBRL Expert Panel - Using Oracle Database as an XBRL Repository**

Hotel Nikko Nikko Ballroom I

3:30 PM – 4:30 PM