XML and Databases

Outline of the talk

- Mapping XML into relational data
- Generating XML using Java and JDBC
- Storing XML
- XML on the Web
- XML support in Oracle
- XML API for databases
- Conclusion

- Mapping XML into relational data
- Generating XML using Java and JDBC
- Storing XML
- XML on the Web
- XML support in Oracle
- XML API for databases
- Conclusion

The database

- We can model the database with a document node and its associated element node:

<?xml version="1.0" ?>
<*myDatabase*>
table$_1$
table$_2$
...
table$_n$
</*myDatabase*>

- Order of tables is immaterial

The table

- Each table of the database is represented by an element node with the records as its children:

<*customer*>
<*record*$_1$>
<*record*$_2$>
...
<*record*$_m$>
</*customer*>

- Again, order of the records is immaterial, since the relational data model defines no ordering on them.

The record

- A record is also represented by an element node, with its fields as children:

<*custRec*>
    $field_1$
    $field_2$
    ...
    $field_m$
</*custRec*>

- The *name* is arbitrary, since the relational data model doesn't define a name for a record type

The field

- A field is represented as an element node with a data node as its only child:

  <*custName* type="*t*">
  d
  </*custName*>

- If *d* is omitted, it means the value of the fields is the empty string.
- The value of *t* indicates the type of the value

Example

```
<?xml version="1.0" ?>
<myDatabase>
    <customers>
        <custRec>
          <custName type="String">Robert Roberts</custName>
          <custAge type="Integer">25</custAge>
        </custRec>
        <custRec>
          <custName type="String">John Doe</custName>
          <custAge type="Integer">32</custAge>
        </custRec>
    </customers>
</myDatabase>
```

- Mapping XML into relational data
- Generating XML using Java and JDBC
- Storing XML
- XML on the Web
- XML support in Oracle
- XML API for databases
- Conclusion

Generating XML from relational data

Step 1 : Set up the database connection

```
// Create an instance of the JDBC driver so that it has
// a chance to register itself
   Class.forName(sun.jdbc.odbc.JdbcOdbcDriver).newInstance();

// Create a new database connection.
  Connection con =
        DriverManager.getConnection(jdbc:odbc:myData, “”, “”);

// Create a statement object that we can execute queries with
  Statement stmt = con.createStatement();
```

Generating XML (cont.)

Step 2 : Execute the JDBC query

```
String query = “Select Name, Age from Customers”;
ResultSet rs = stmt.executeQuery(query);
```

Generating XML (cont.)

Step 3 : Create the XML!

```
StringBuffer xml = “<?xml
    version=‘1.0’?><myDatabase><customers>”;

while (rs.next()) {
    xml.append(“<custRec><custName>”);
    xml.append(rs.getString(“Name”));
    xml.append(“</custName><custAge>”);
    xml.append(rs.getInt(“Age”));
    xml.append(“</custAge></custRec>”);
}
xml.append(“</customers></myDatabase>”);
```

- Mapping XML into relational data
- Generating XML using Java and JDBC
- Storing XML
- XML on the Web
- XML support in Oracle
- XML API for databases
- Conclusion

Storing XML in relational tables

Step 1 : Set up the parser

```
StringReader stringReader = new StringReader(xmlString);
InputSource inputSource = new InputSource(stringReader);
DOMParser domParser = new DOMParser();
domParser.parse(inputSource);
Document document = domParser.getDocument();
```

Storing XML (cont.)

Step 2 : Read values from parsed XML document

```
NodeList nameList = doc.getElementsByTagName(“custName”);
NodeList ageList = doc.getElementsByTagName(“custAge”);
```

Storing XML (cont.)

Step 3 : Set up database connection

```
Class.forName(sun.jdbc.odbc.JdbcOdbcDriver).newInstance();
Connection con =
        DriverManager.getConnection(jdbc:odbc:myDataBase, “”, “”);
Statement stmt = con.createStatement();
```

Storing XML (cont.)

Step 4 : Insert data using appropriate JDBC update query

```
String sql = “INSERT INTO Customers (Name, Age) VALUES (?,?)”;
PreparedStatement pstmt = conn.prepareStatement(sql);
int size = nameList.getLength();
for (int i = 0; i < size; i++) {
        pstmt. setString(1, nameList.item(i).getFirstChild().getNodeValue());
        pstmt.setInt(2, ageList.item(i).getFirstChild().getNodeValue());

        pstmt.executeUpdate(sql);
}
```

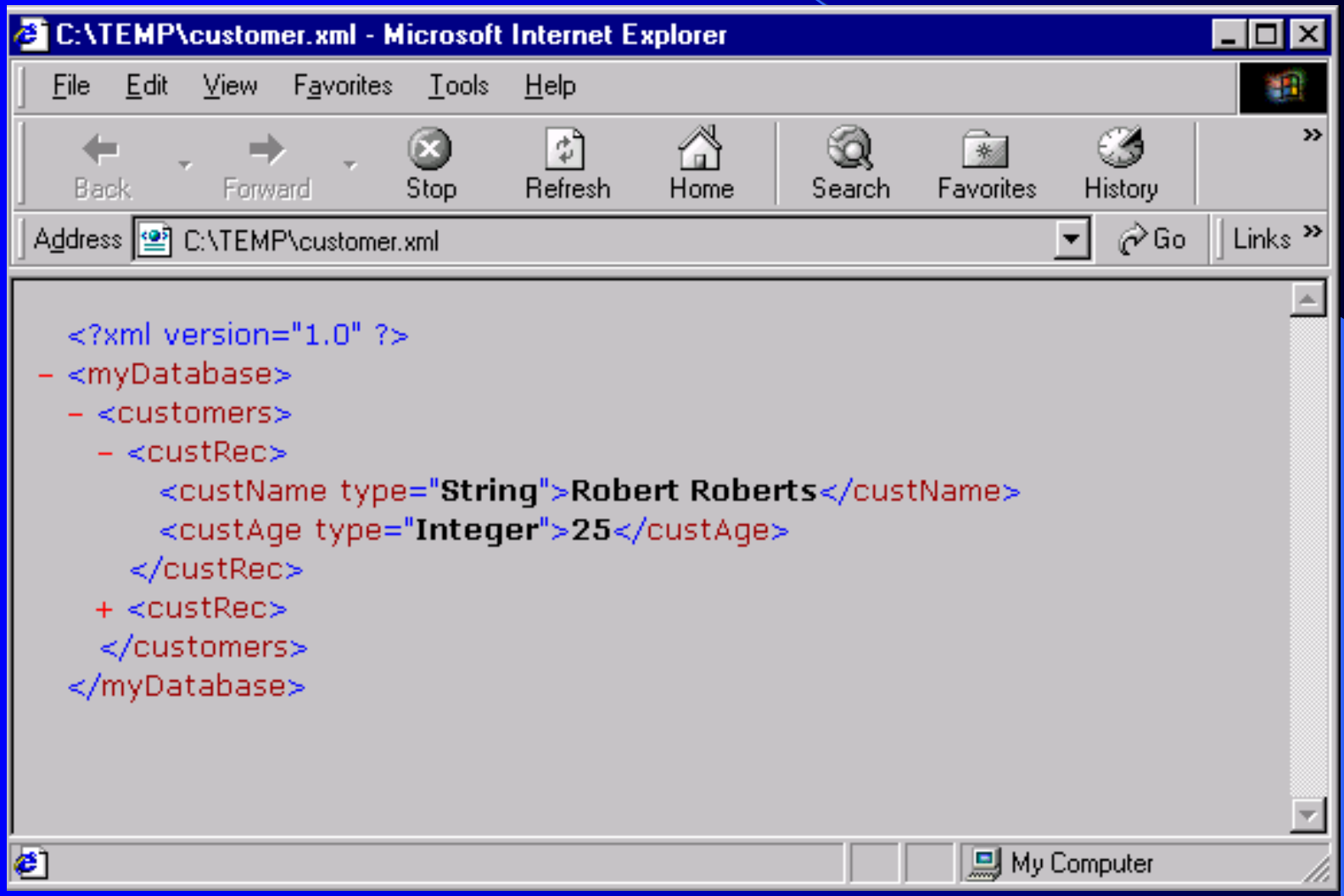
- Mapping XML into relational data
- Generating XML using Java and JDBC
- Storing XML
- XML on the Web
- XML support in Oracle
- XML API for databases
- Conclusion

XML on the Web (Servlets)

```
public void doGet(HttpServletRequest req, HttpServletResponse
    resp)
{
    resp.setContentType("text/xml");
     PrintWriter out = new PrintWriter(resp.getOutputStream());
    … generate XML here, as before…
    out.println(xmlGenerated);
    out.flush();
    out.close();
}
```

- Appropriate XSL can be inserted for display

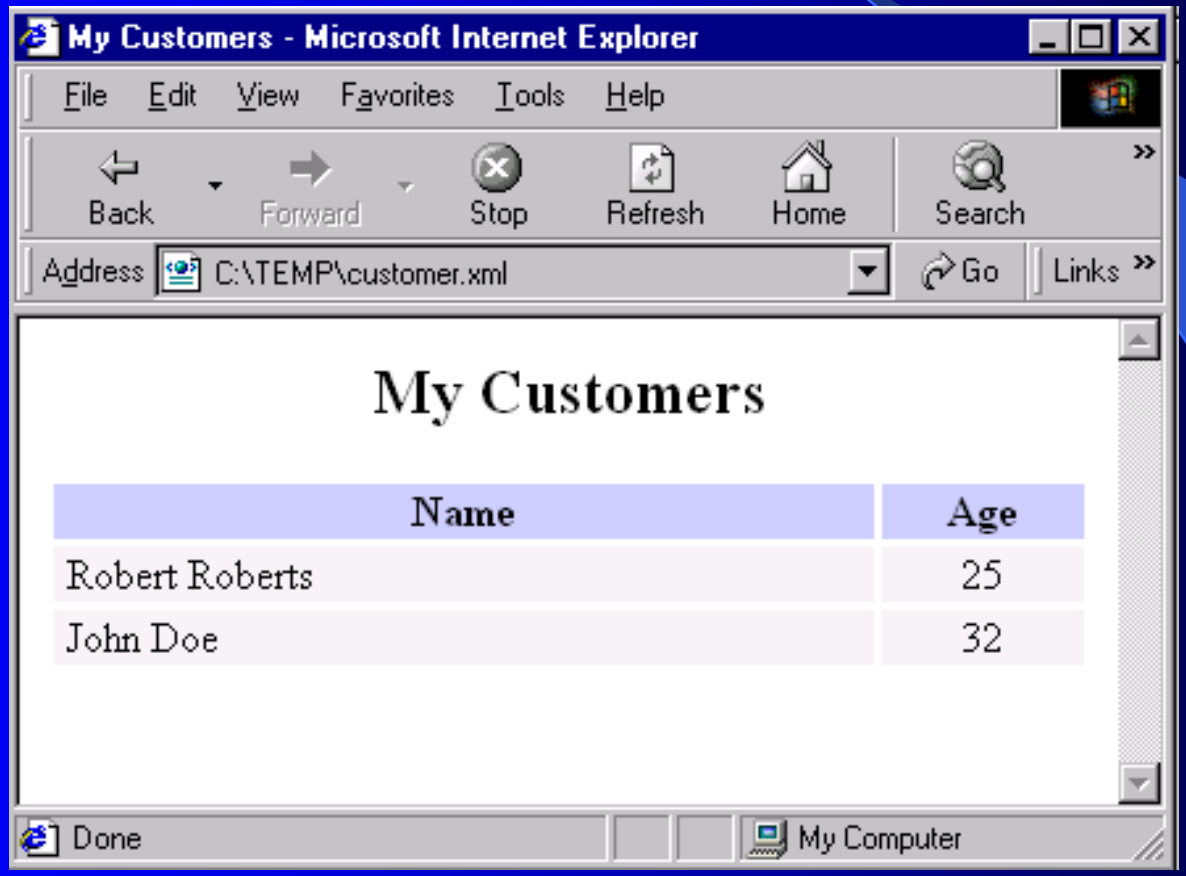
XML in IE 5.0


C:\TEMP\customer.xml - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Forward Stop Refresh Home Search Favorites History
Address C:\TEMP\customer.xml
Go Links
<?xml version="1.0" ?>
- <myDatabase>
- <customers>
- <custRec>
<custName type="String">Robert Roberts</custName>
<custAge type="Integer">25</custAge>
</custRec>
+ <custRec>
</customers>
</myDatabase>
My Computer

Let's insert the XSL…

```
<?xml version="1.0" ?>
<?xml-stylesheet type="text/xsl"
    href="http://myServer/Customer.xsl"?>
<myDatabase>
    <customers>
        <custRec>
          <custName type="String">Robert Roberts</custName>
          <custAge type="Integer">25</custAge>
        </custRec>
        … other records here …
    </customers>
</myDatabase>
```

XML with XSL in IE 5.0



- Mapping XML into relational data
- Generating XML using Java and JDBC
- Storing XML
- XML on the Web
- XML support in Oracle
- XML API for databases
- Conclusion

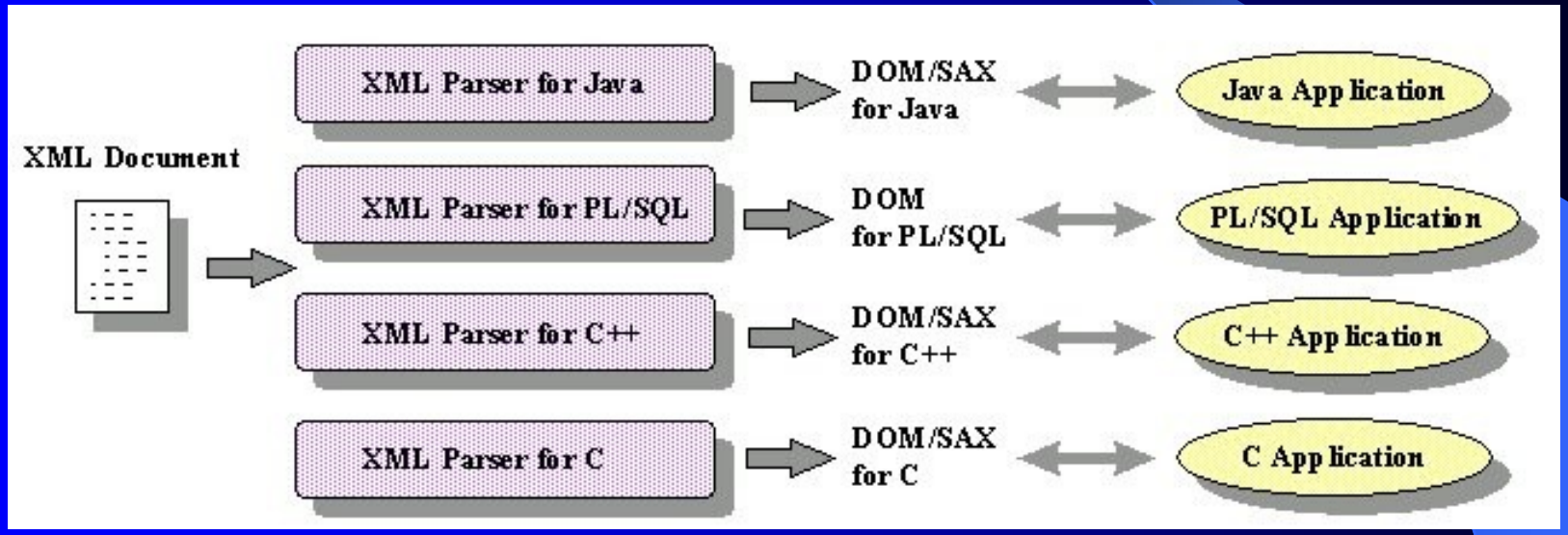
XML support in Oracle

- Oracle8*i* and *inter*Media
- XML Parsers and XSL Processors (Java, C, C++, and PL/SQL)
- XML Class Generators (Java and C++)
- XML SQL Utility for Java
- XSQL Servlet

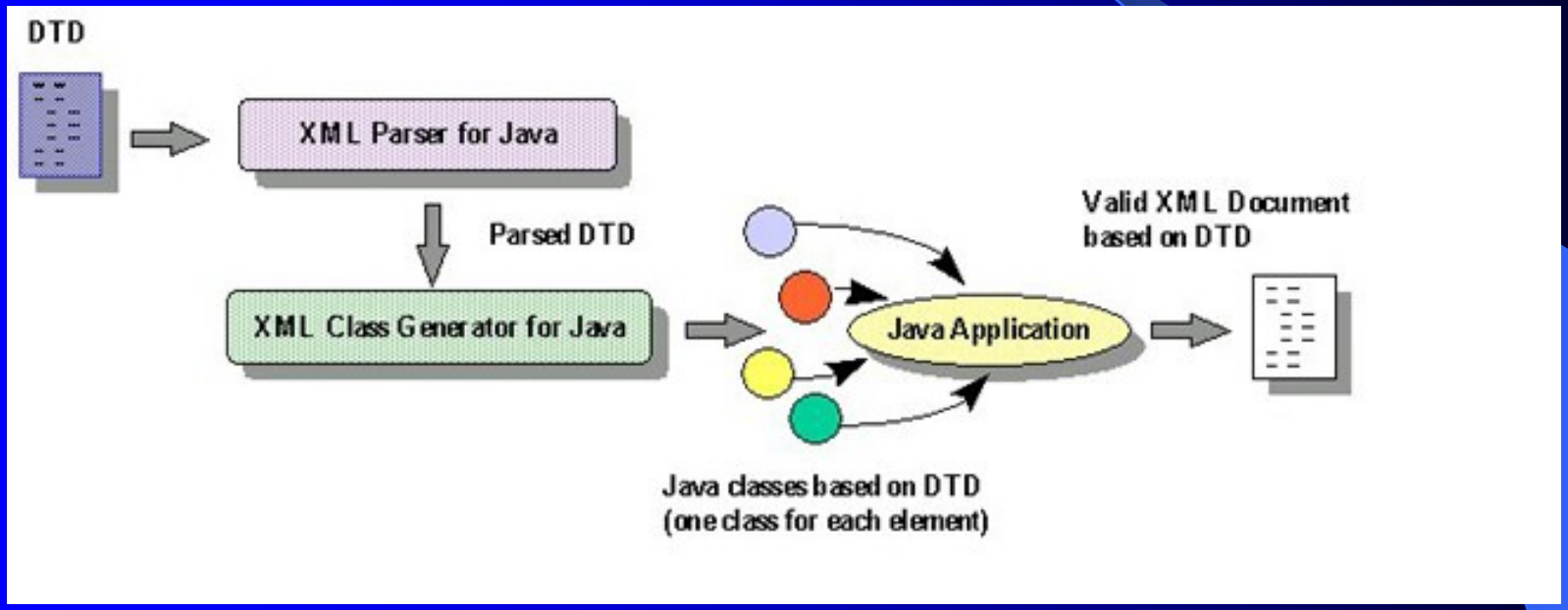
Oracle8*i* and *inter*Media

- run Oracle XML components and applications inside the database using JServer - Oracle8*i*'s built-in JVM
- *inter*Media Text allows queries such as find "Oracle WITHIN title" where "title" is a section of the XML document

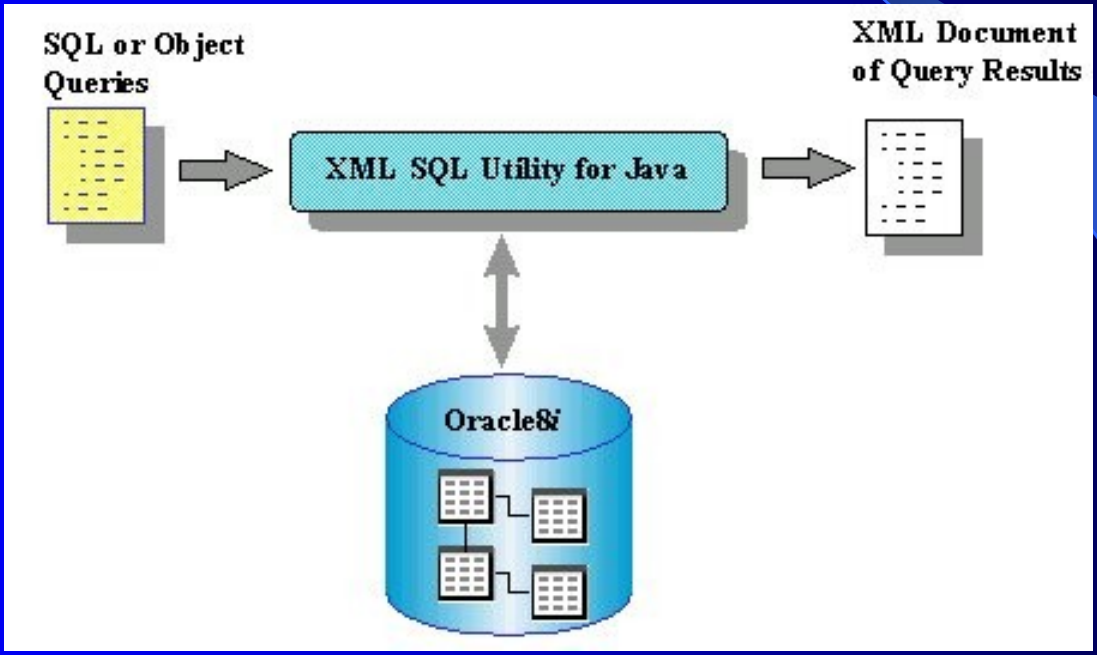
XML Parsers in Oracle



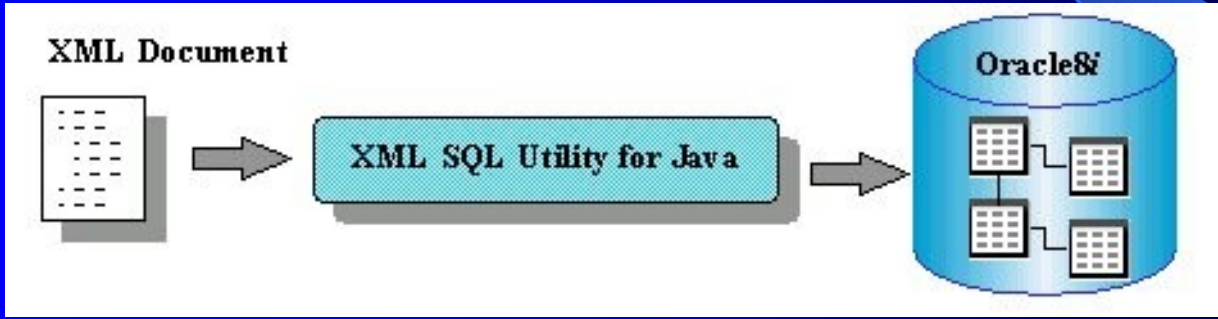
XML Class Generator for Java



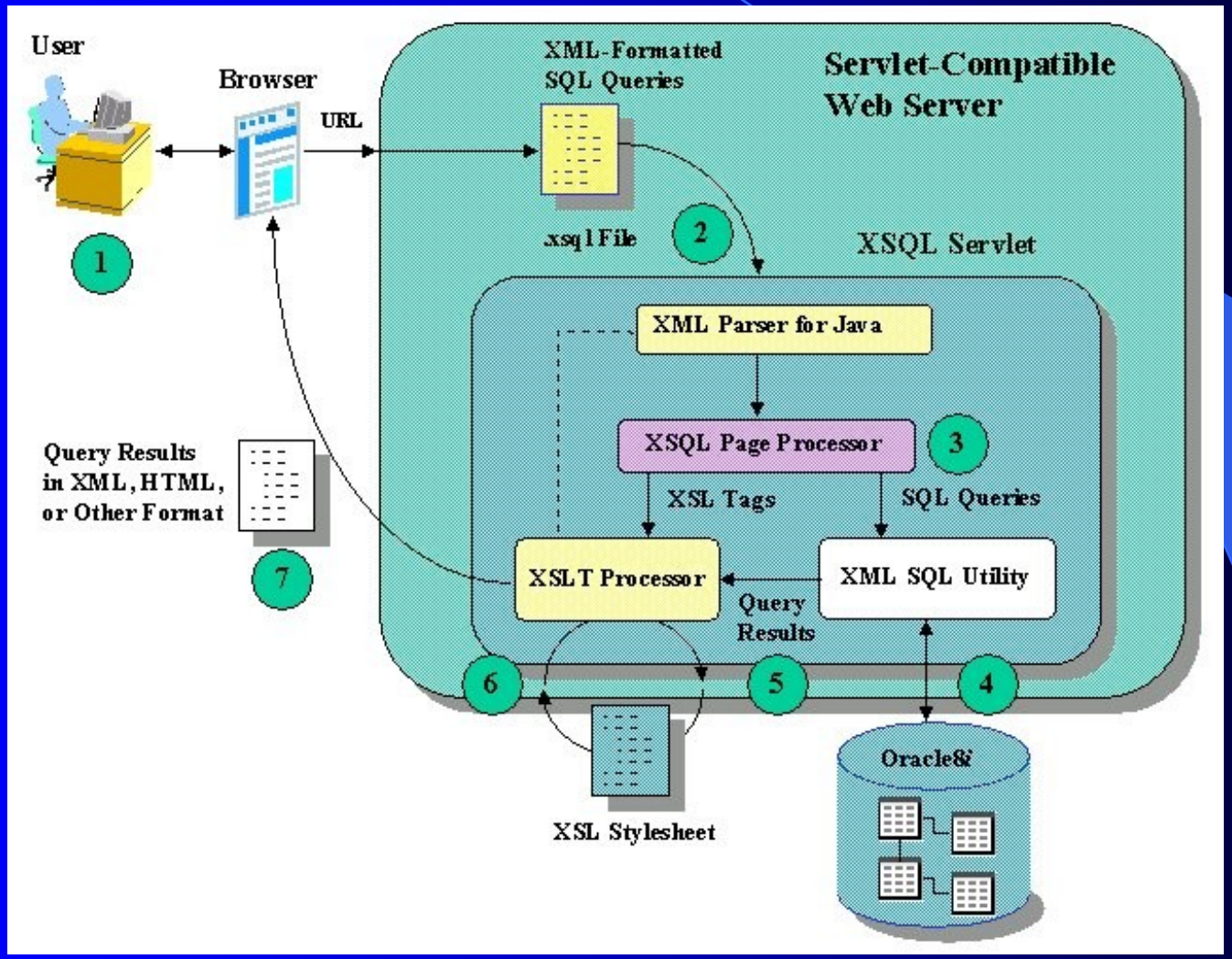
XML SQL Utility for Java
(Generating XML)



XML SQL Utility for Java
(Inserting XML)



XSQL Servlet

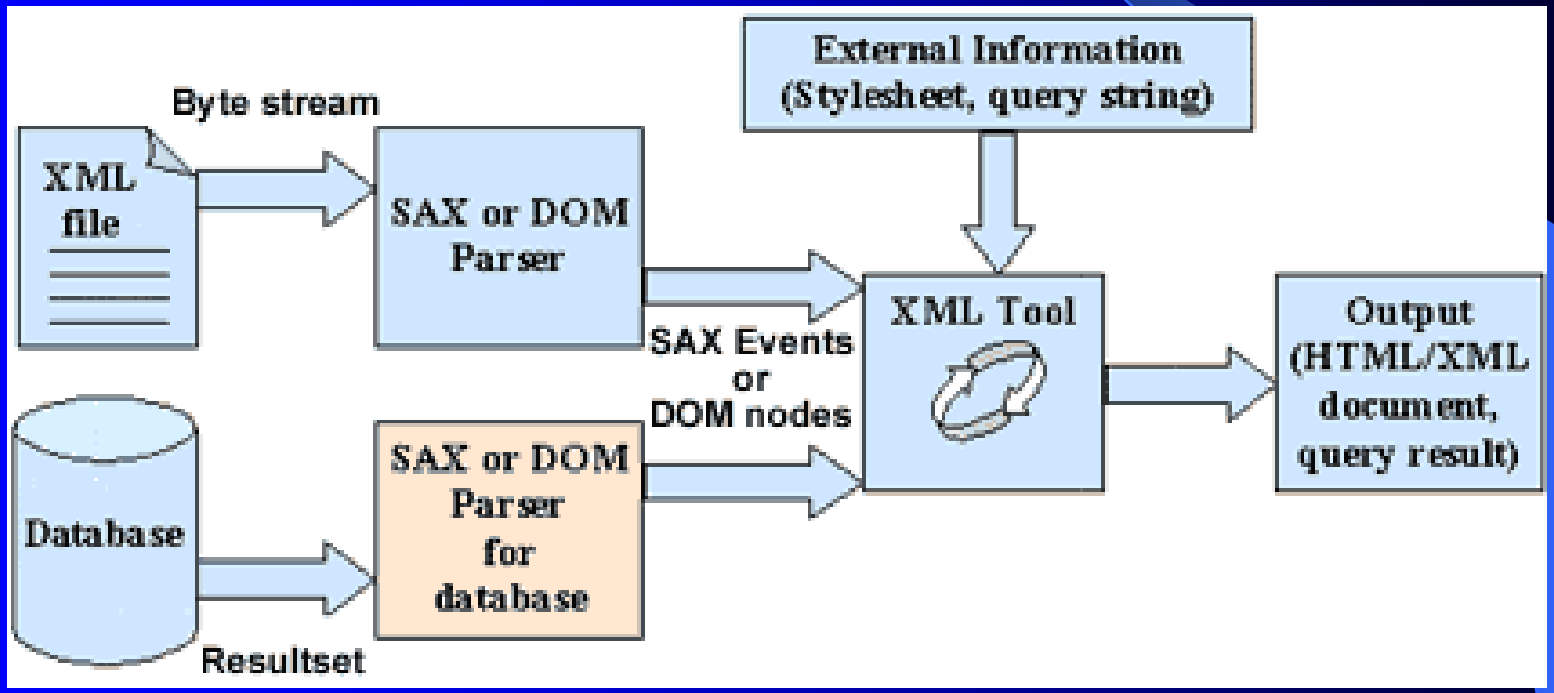


- Mapping XML into relational data
- Generating XML using Java and JDBC
- Storing XML
- XML on the Web
- XML support in Oracle
- XML API for databases
- Conclusion

XML API for databases
Ramnivas Laddad, JavaWorld, Feb 2000

- blend the power of a database with the features of XML
- most XML tools work with the SAX or DOM API
- implement the same APIs directly over a database, enabling XML tools to treat databases as if they were XML documents. That way, we can obviate the need of converting a database.

XML API with database



- Mapping XML into relational data
- Generating XML using Java and JDBC
- Storing XML
- XML on the Web
- XML support in Oracle
- XML API for databases
- Conclusion

Conclusion

- XML to relational data is easy (?)
- Lots of tools, support for XML in databases emerging… research, as well as commercial!

- However, there are still a lot of unresolved issues… we'll look at them in XML and Challenges.