IBM Software Group

IMS19

IMS v9 XML-DB

Christopher Holtz

IMS Development

Silicon Valley Laboratory

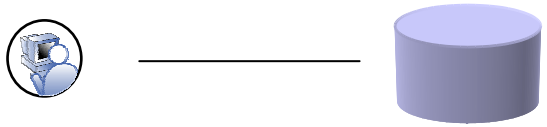
IBM Corporation

Overview

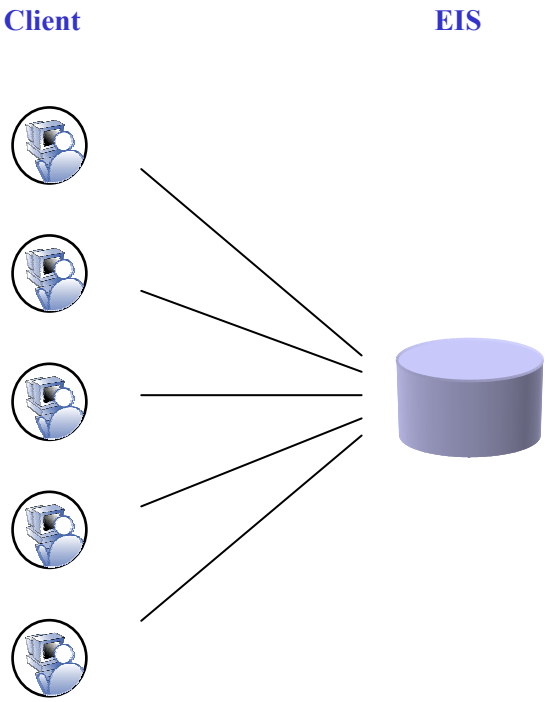
- Introduction
  - What is IMS XML-DB?
  - What is XML?
  - What are XML Schemas?
- IMS XML-DB Methodology (XML Visualization)
  - Decomposed Storage
  - Intact Storage
- IMS Java / XML-DB Tooling
  - Metadata Generation
- IMS XML-DB Java Implementation
  - SQL UDF Interface
  - Future Direction

Client Server Architecture

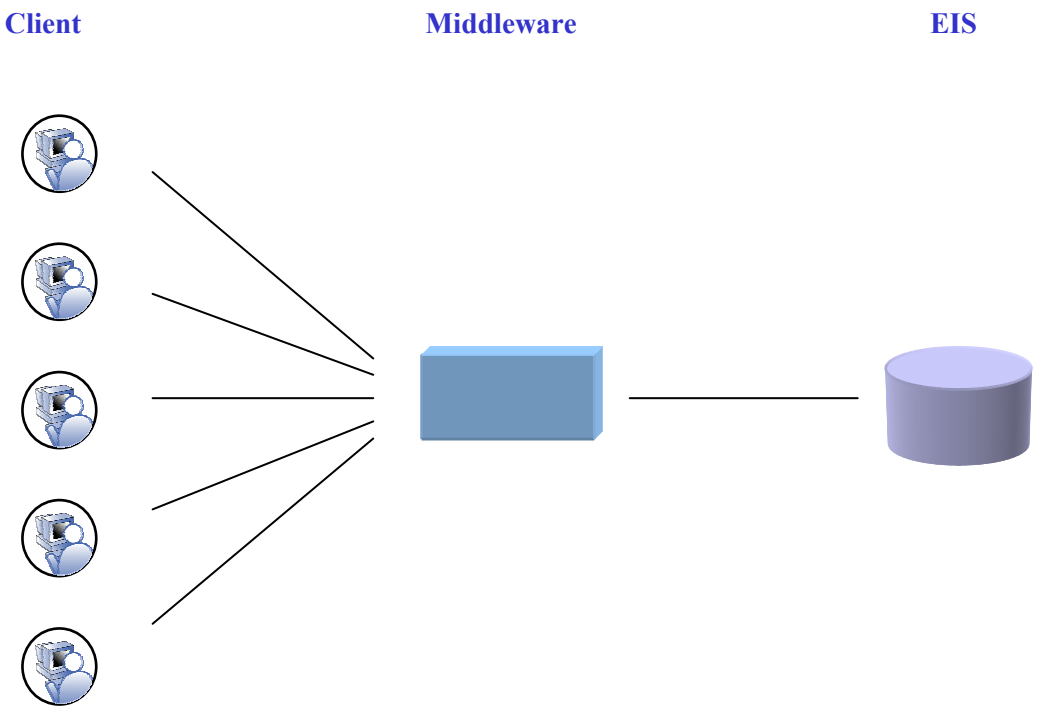
Client **EIS**



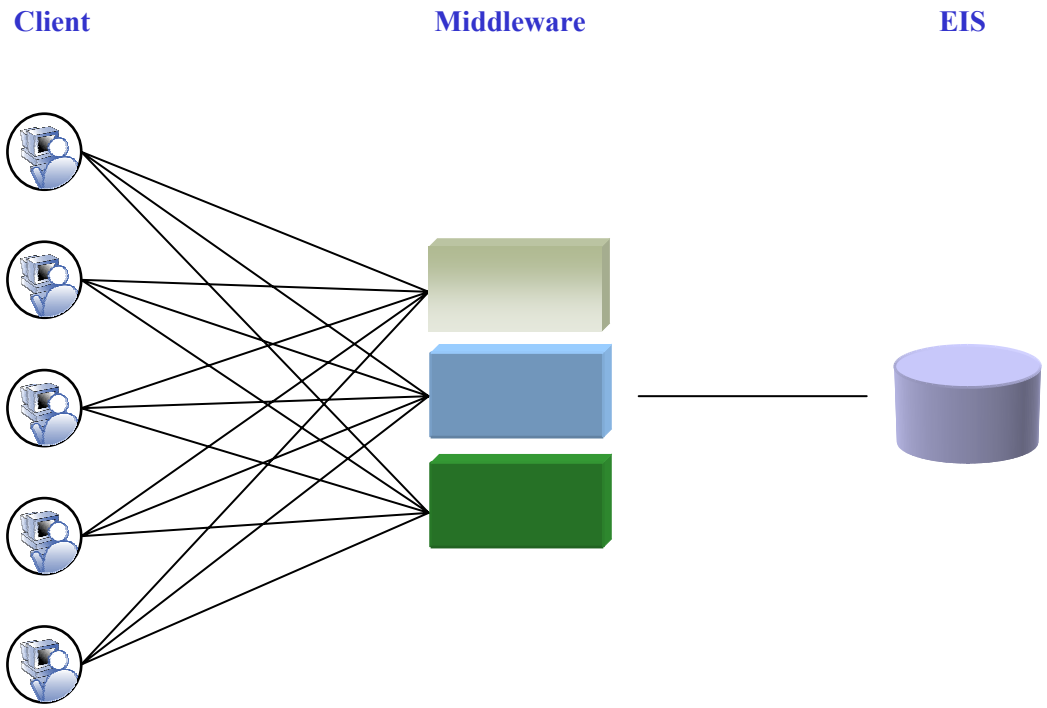
Client Server Architecture



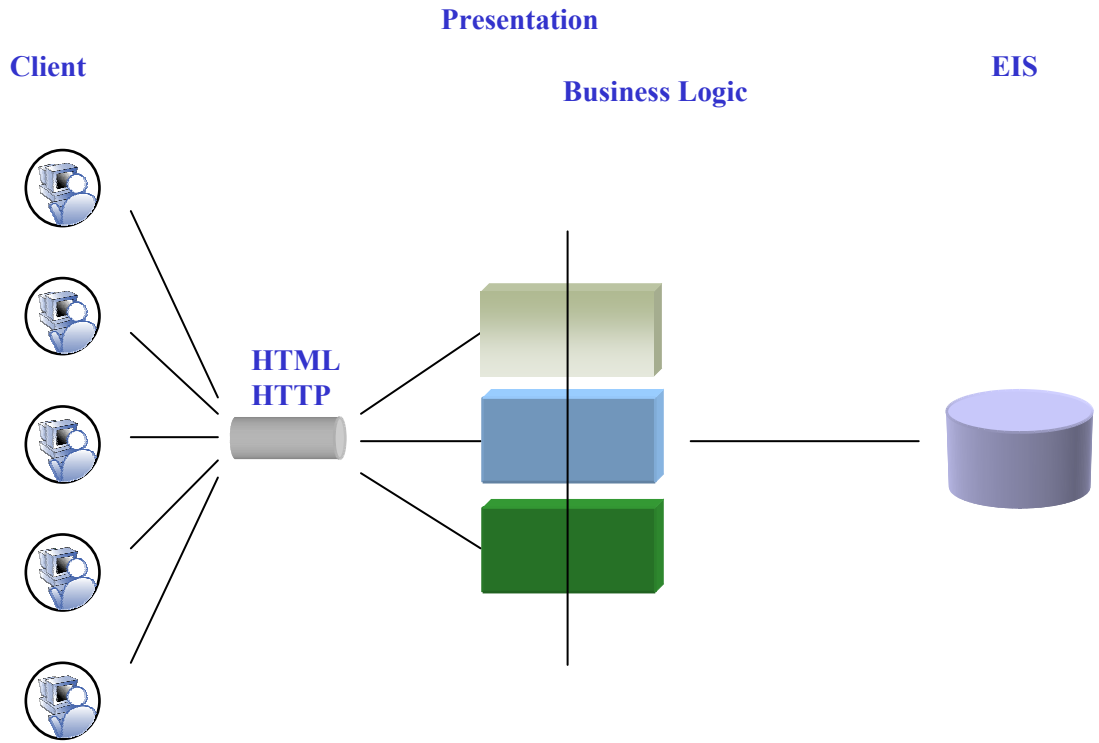
3-tier Architecture



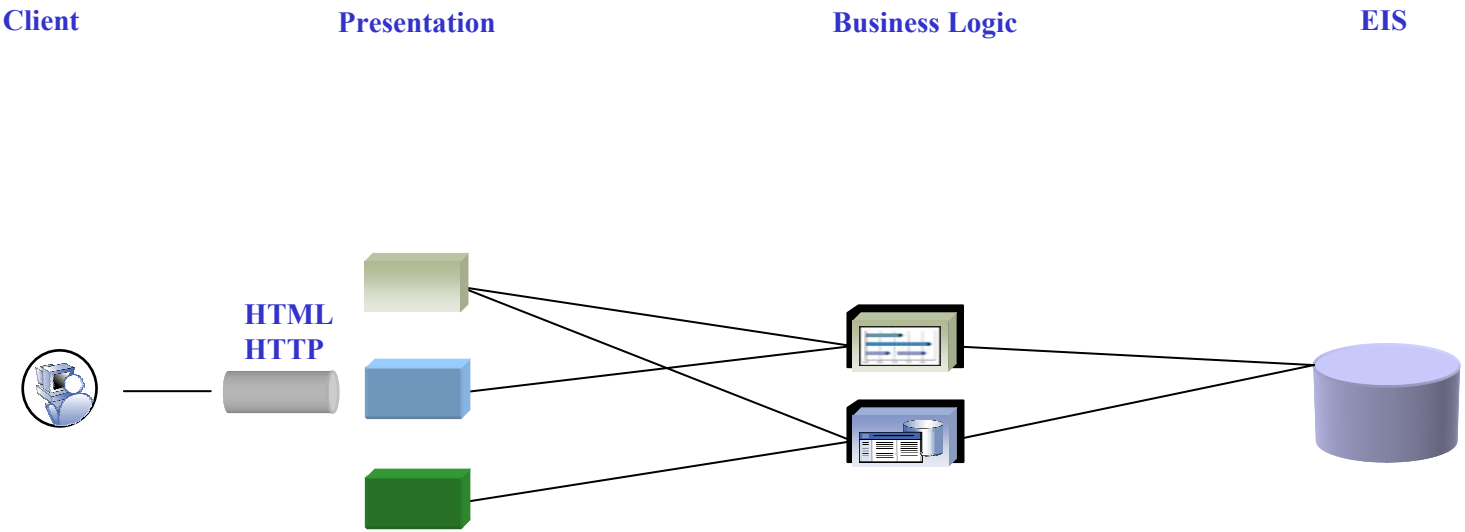
3-tier Architecture



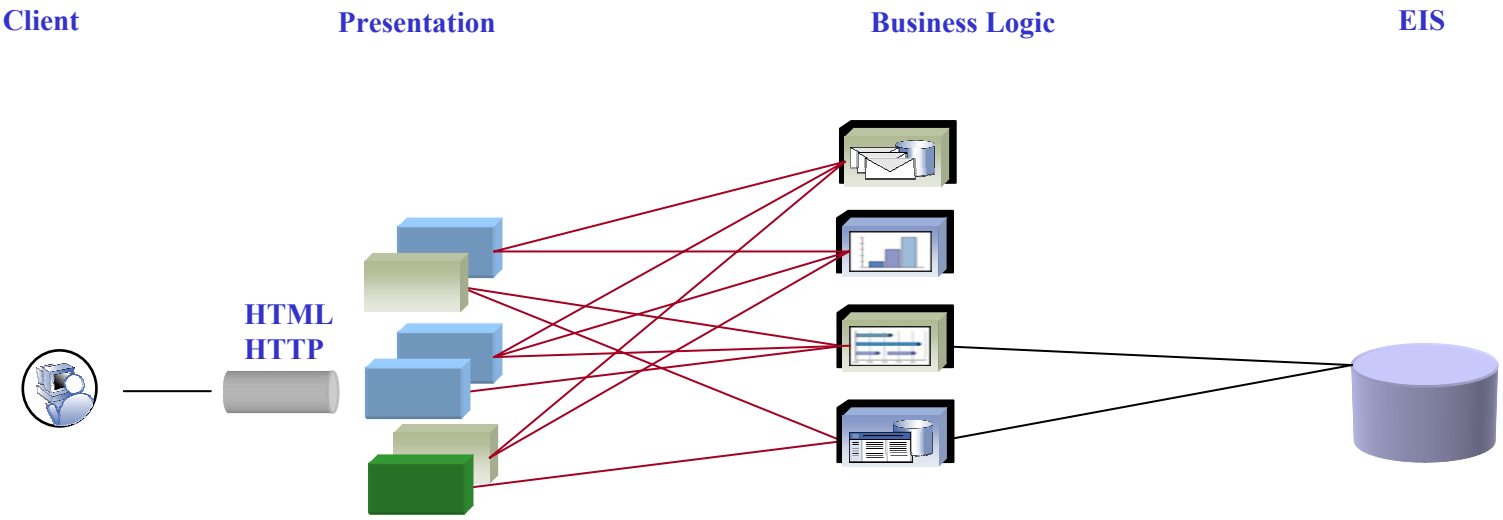
3-tier Architecture



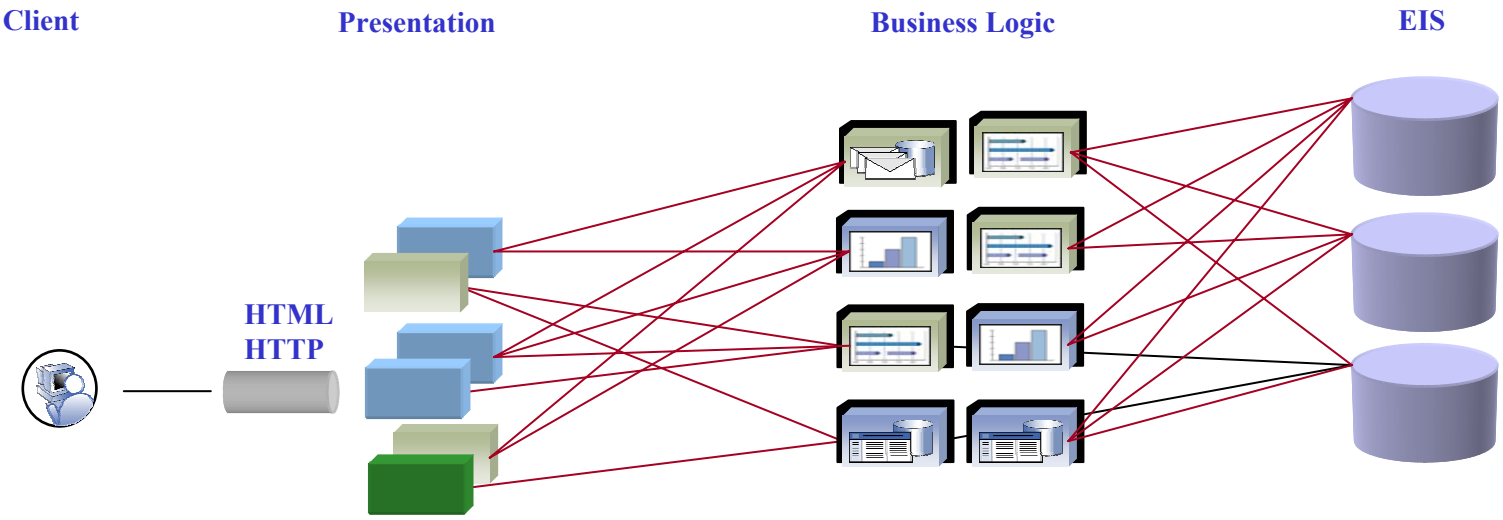
4-tier Architecture



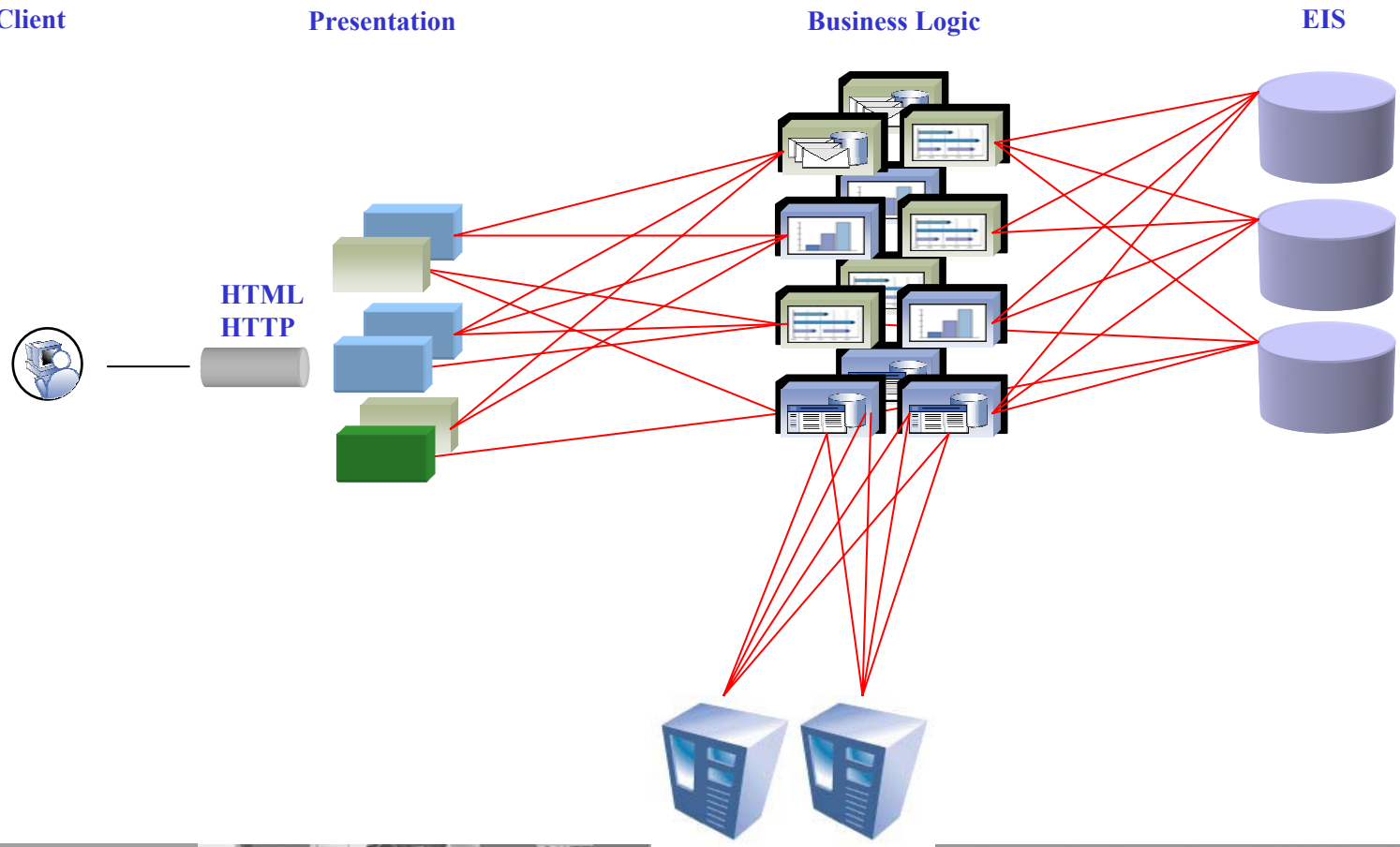
4-tier Architecture



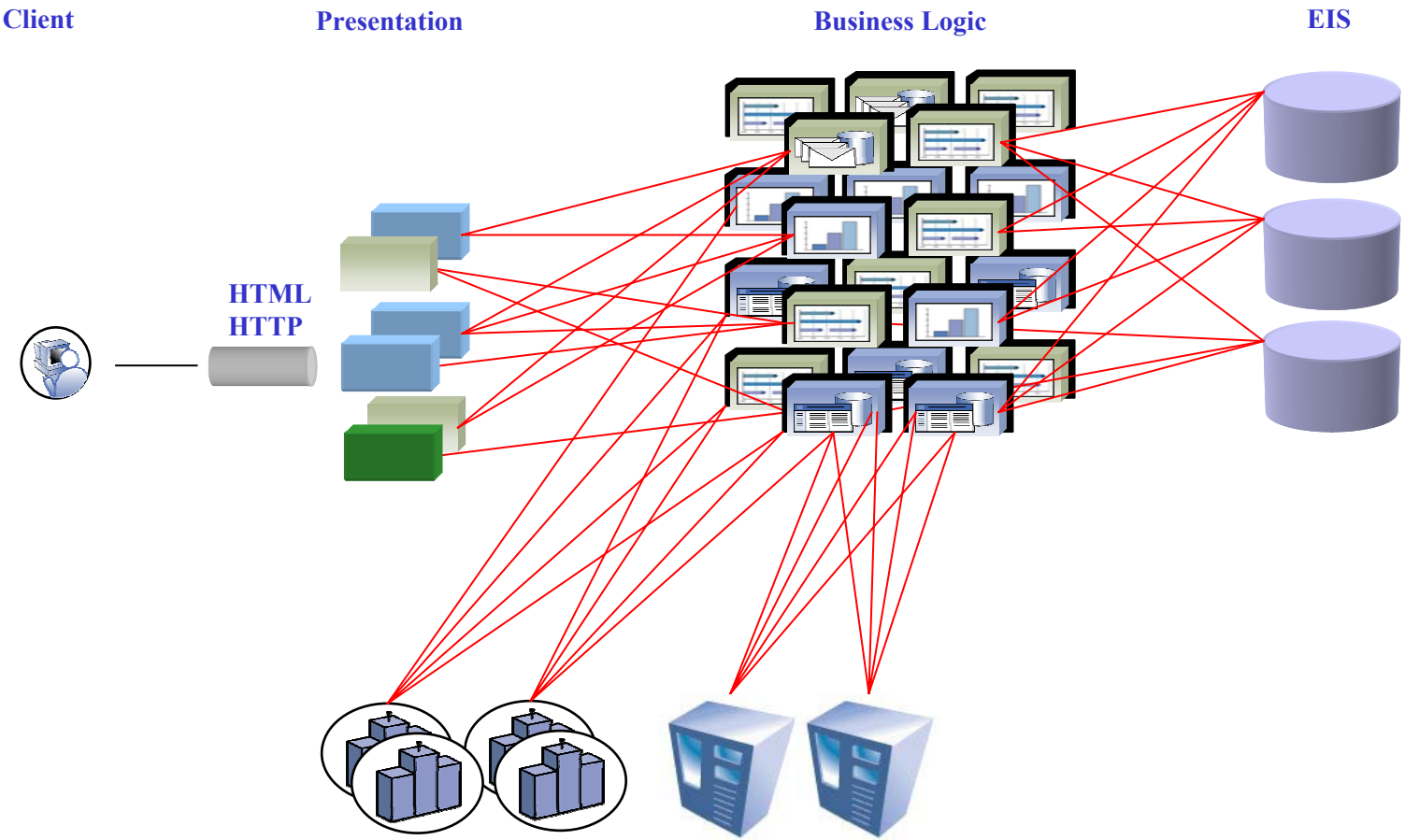
4-tier Architecture



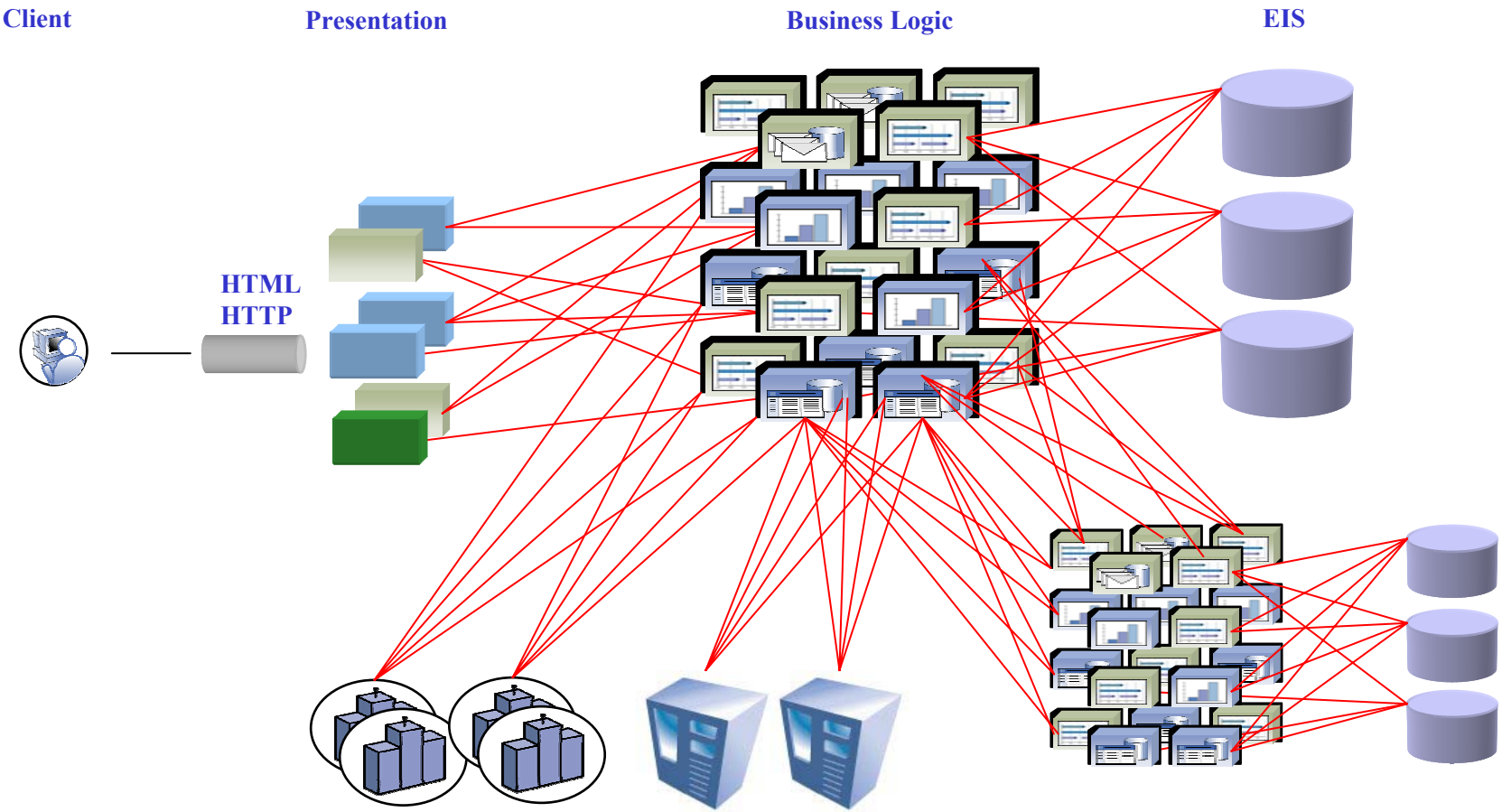
4-tier Architecture



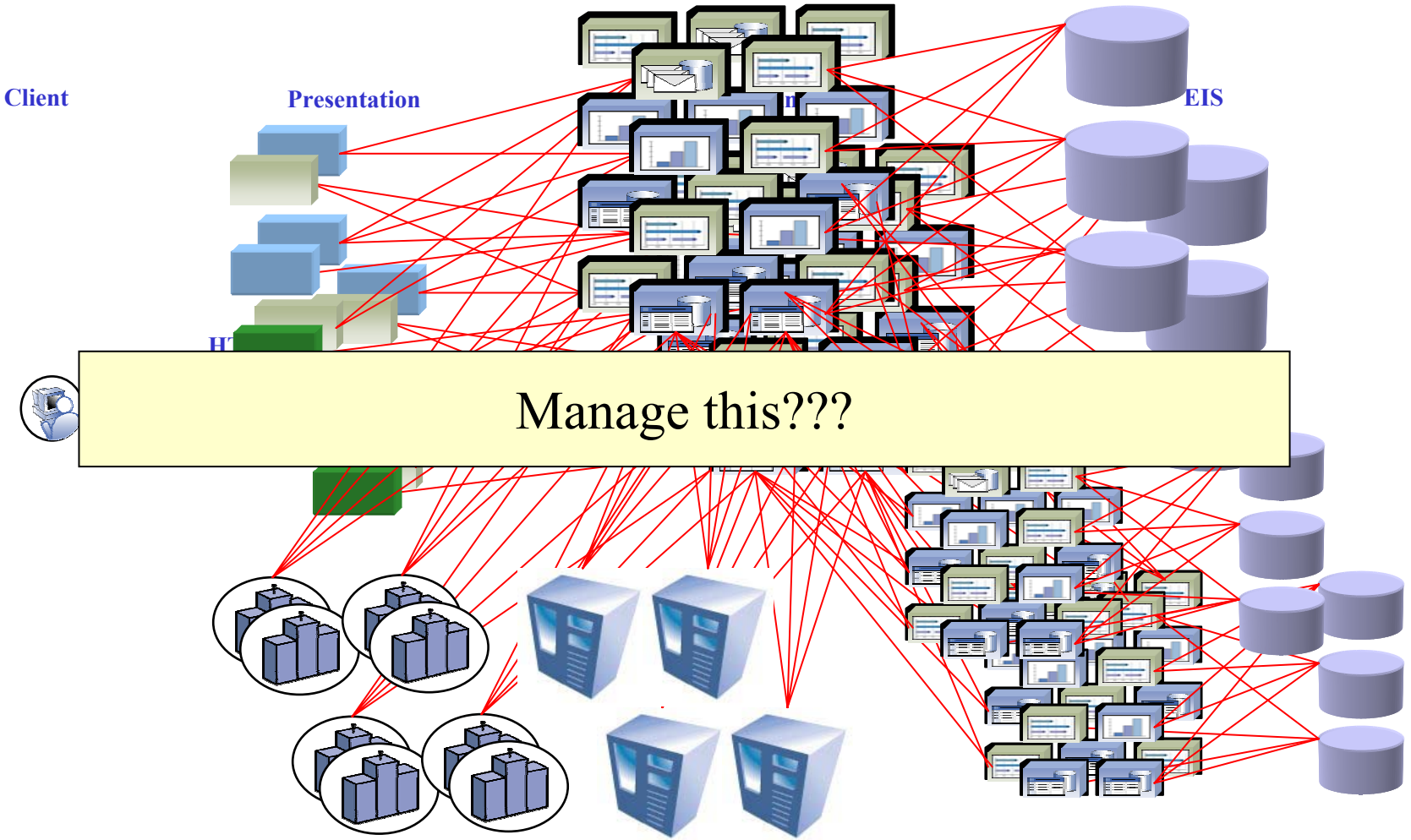
4-tier Architecture



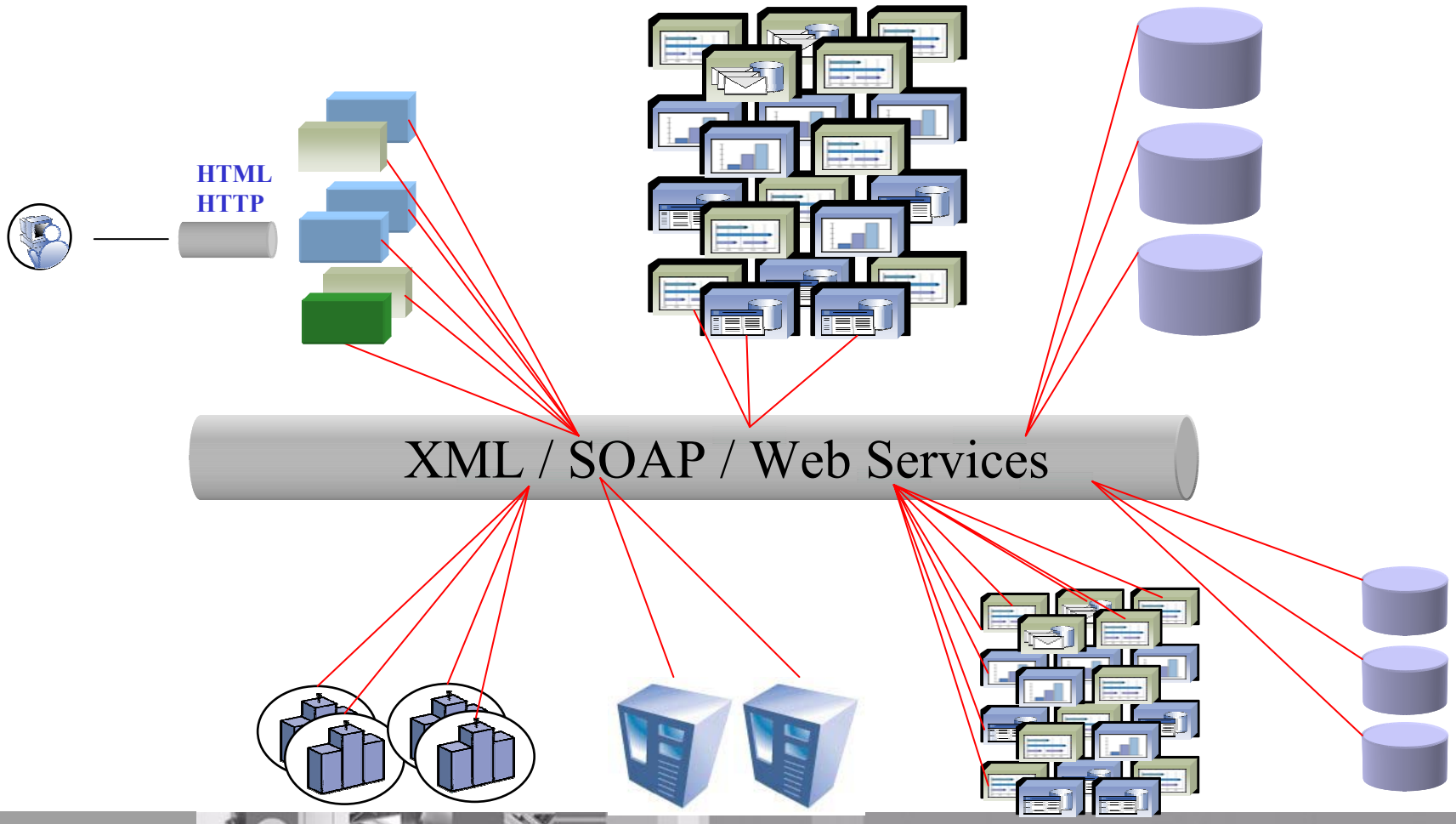
4-tier Architecture



4-tier Architecture

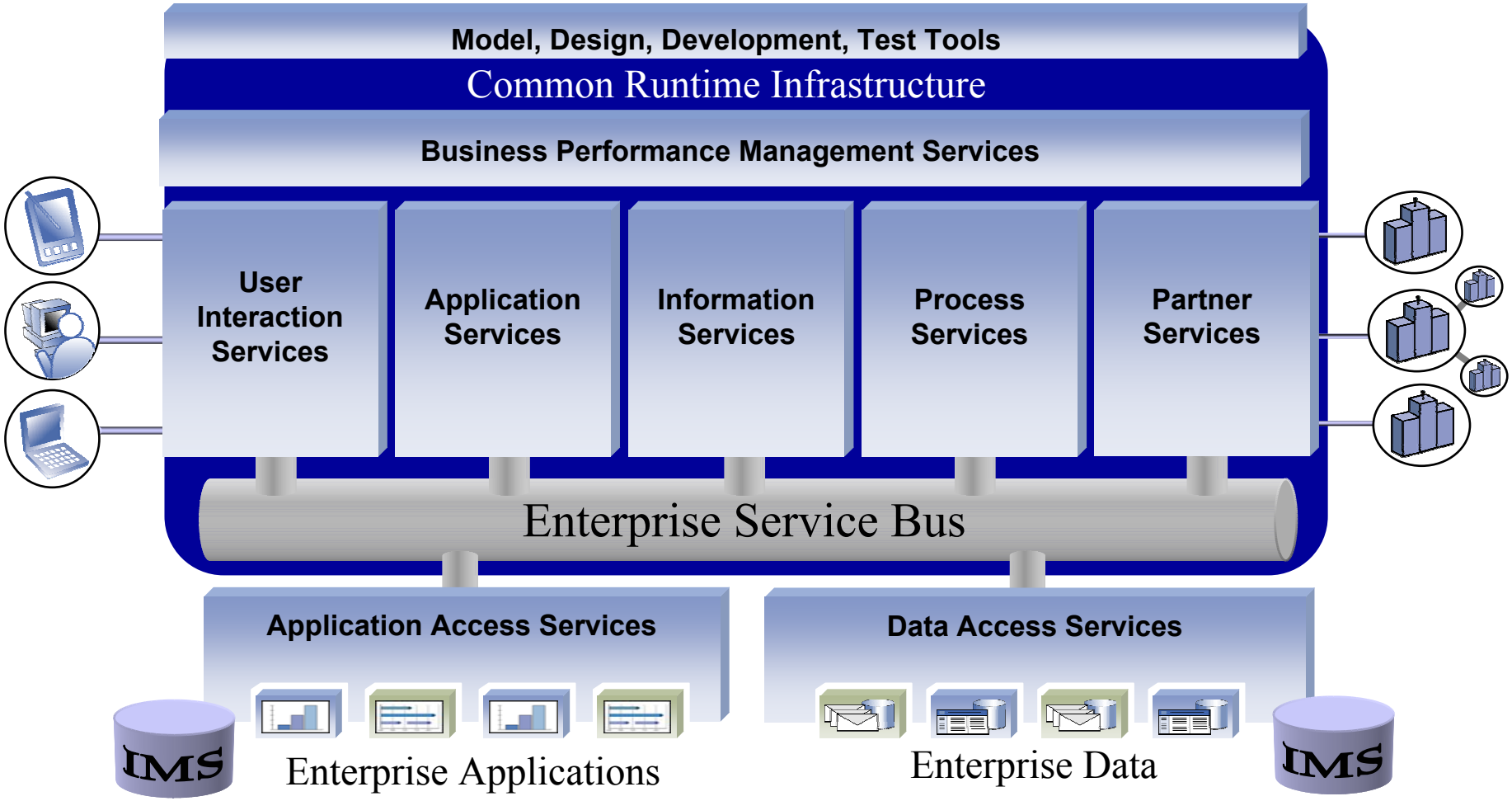


XML / SOAP / Web Services



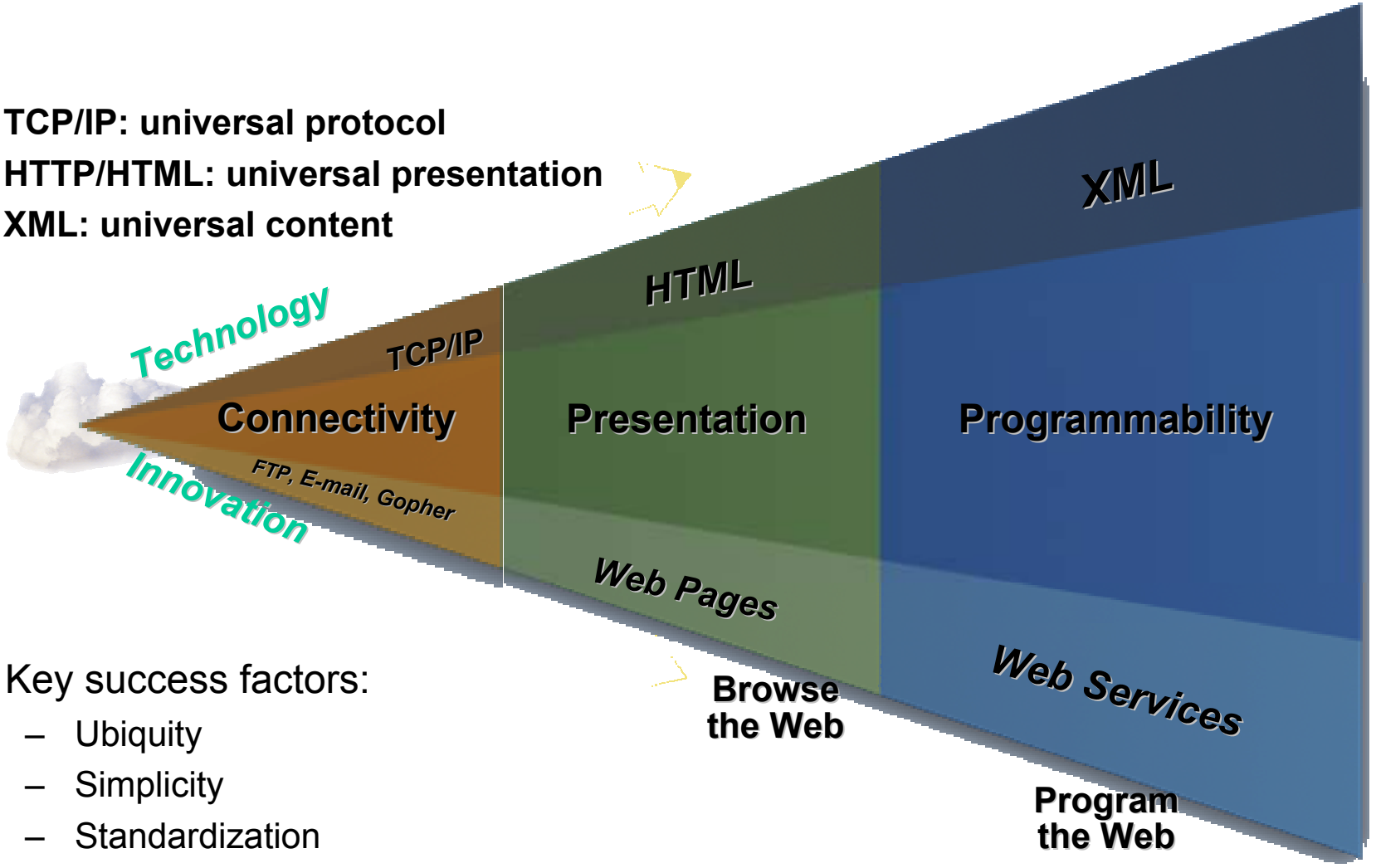
IBM Service Oriented Architecture

ESB Is a Defining Element



The Path to Web Services

- **TCP/IP: universal protocol**
- **HTTP/HTML: universal presentation**
- **XML: universal content**



- Key success factors:
  - Ubiquity
  - Simplicity
  - Standardization

What is IMS XML-DB

- New for IMS version 9.1
- A methodology for storing and retrieving XML documents into and out of standard IMS databases
  - Language Independent Design
  - XML Schema Metadata (Structural Metadata)
  - DL/I Metadata (Physical Metadata)
  - Two storage types
- Java enablement of XML-DB using an extended IMS Java JDBC interface

Why use IMS XML-DB

- A World-wide movement towards XML as the standard data interchange language.

- Retrieve existing IMS data in standard, easily exchangeable XML format
- Store, Index, Search and Retrieve valid new XML documents into new or existing IMS databases

- 35 years of storage and management of Hierarchical data
- 35 years of performance, stability and reliability

What is XML

- A Standardized, Simple, and Self-Describing Markup Language for documents containing structured or semi-structured information.

```
<?xml version="1.1"?>
<Presentation>
   <title>XML-DB</title>
   <length>60</length>
   <presenter>
      <lastName>Holtz</lastName>
      <firstName>Christopher</firstName>
   </presenter>
   <Comments session="e-business Design Review">
      <comment>Loved It</comment>
      <comment>Can't wait for GA</comment>
   </Comments>
</Presentation>
```

Why XML…

- Standard Internet Data Exchange Format
  - Self-Describing
  - Handles encoding

    <xml? version="1.1" encoding="ebcdic-cp-us"?>
  - Handles byte ordering

    <OrderNumber>**110203**</OrderNumber>
  - Human Legible?
  - Easily Parsed

  - **Standard!**

What is XML

- Data-centric
  - Highly structured
  - Limited size and strongly typed data elements
  - Order of elements generally insignificant
  - Invoices, purchase orders, etc.

- Document-centric
  - Loosely structured
  - Unpredictable sizes with mostly character data
  - Order of elements significant
  - Newspaper articles, manuals, etc.

Well formed vs. Valid XML Document

- Well formed – Obeys the XML Syntax Rules
  - must begin with the XML declaration
  - must have one unique root element
  - all start tags must match end-tags
  - XML tags are case sensitive
  - all elements must be closed
  - all elements must be properly nested
  - all attribute values must be quoted
  - XML entities must be used for special characters
- Valid – Conforms to a specific XML Schema

The XML Schema Definition Language

An XML language for defining the legal building blocks of a valid XML document

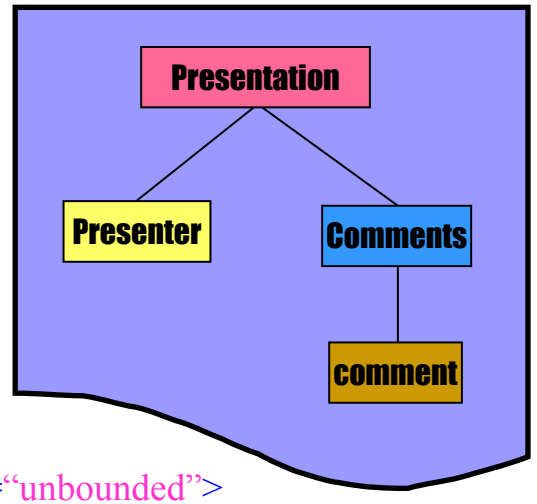
An XML Schema:

- defines elements and attributes that can appear in a document
- defines which elements are child elements
- defines the order and number of child elements
- defines whether an element is empty or can include text
- defines data types for elements and attributes
- defines default and fixed values for elements and attributes

Defines an agreed upon communication contract for exchanging XML documents

XML Schema Example

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
        xmlns="http://www.myNamespace.net"
        targetNamespace="http://www.myNamespace.net"
        elementFormDefault="qualified">

  <xsd:element name="Presentation">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element name="title" type="xsd:string"/>
        <xsd:element name="length" type="xsd:integer"/>
        <xsd:element name="Presenter" minOccurs="0" maxOccurs="unbounded">
          <xsd:element name="lastName" type="xsd:string"/>
          <xsd:element name="firstName" type="xsd:string"/>
        </xsd:element>
        <xsd:element name="Comments" minOccurs="0" maxOccurs="unbounded">
          <xsd:element name="comment" minOccurs="0" maxOccurs="unbounded" type="xsd:string"/>
          <xsd:attribute name="session" type="xsd:string"/>
      …
</xsd:schema>
```

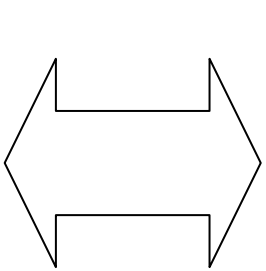


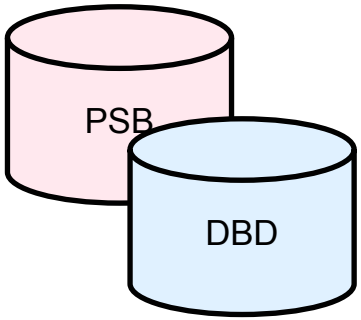
XML Storage in IMS

- Natural mapping between hierarchic XML data and hierarchic IMS database definitions.

XML Schema

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
        xmlns:ims="http://www.ibm.com/ims"
        xmlns="http://www.ibm.com/ims/PSBName/PCBName"
        targetNamespace="http://www.ibm.com/ims/PSBName/PCBName"
        elementFormDefault="qualified">

  <xsd:annotation>
    <xsd:appinfo>
      <ims:DLI mode="store" PSB="AUTPSB11" PCB="AUTOLPCB"
            dsg="DATASETG" meanLength="1000" numDocs="100"/>
    </xsd:appinfo>
  </xsd:annotation>

  <xsd:element name="A">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element name="field1" type="xsd:int"/>
        <xsd:element name="field2">
          <xsd:simpleType>
            <xsd:restriction base="xsd:string">
              <xsd:maxLength value="30"/>
            </xsd:restriction>
  ...
```





It's the Metadata, Stupid!

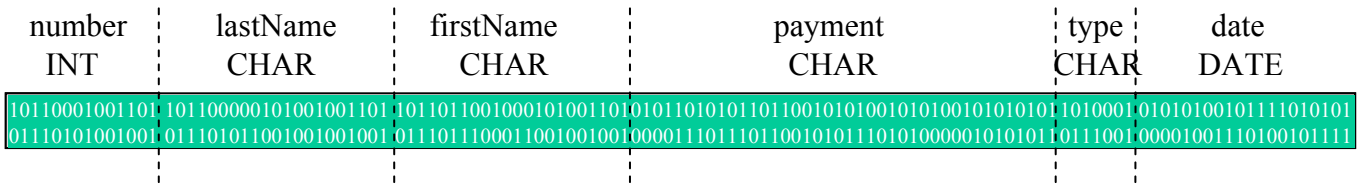
- Physical Metadata
  - Segment Sizes
  - Segment Hierarchy (*field relationships – 1-to-1, 1-to-n*)
  - DBD Defined Fields
  - Application Defined Fields
  - Field Type, Type Length, Byte Ordering, Encoding, etc.
  - Offer Field/Segment Renaming *(lift 8 char restriction)*
- Structural Metadata
  - XML layout for fields (*field relationships must still match*)
  - Element vs. Attribute (*names must match*)
  - Type Restrictions, Enumerations, etc.

Defined in DBD

Defined in Copylibs (IMS Java)

Defined in XML Schema

It's the Metadata, Stupid!



Defined in DBD

Defined in Copylibs (IMS Java)

Defined in XML Schema

```
<PurchaseOrder number="113246 ">
   <lastName> Holtz          </lastName>
   <firstName> Christopher</firstName>
   <date>10/21/2003 </date>
   <payment type="MC "> 5414 2263 4895 1145 </payment>
```

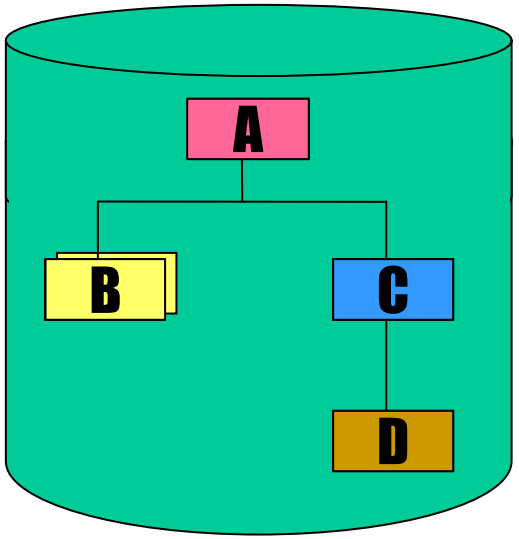
Decomposed XML Retrieval in IMS

Composed XML

```
<A>
  <f1> [redacted] </f1>
  <f2>           </f2>
  <f3>           </f3>
  <B>
    <f4> [redacted] </f4>
    <f5>           </f5>
  </B>
  <B>
    <f4> [redacted] </f4>
    <f5>           </f5>
  </B>
  <C>
    <f6> [redacted] </f6>
    <f7>           </f7>
    <D>
      <f8> [redacted] </f8>
      <f9>           </f9>
    </D>
  </C>
</A>
```

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
        xmlns:ims="http://www.ibm.com/ims"
        xmlns="http://www.ibm.com/ims/PSBName/PCBName"
        targetNamespace="http://www.ibm.com/ims/PSBName/PCBName"
        elementFormDefault="qualified">

  <xsd:annotations go here/>

  <xsd:element name="A">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element name="field1" type="xsd:int"/>
        <xsd:element name="field2">
          <xsd:simpleType>
  ...
  <xsd:element name="B">
    <xsd:complexType>
  ...
  <xsd:element name = "C">
    <xsd:complexType>
  <xsd:element name = "D">
    <xsd:complexType>
```

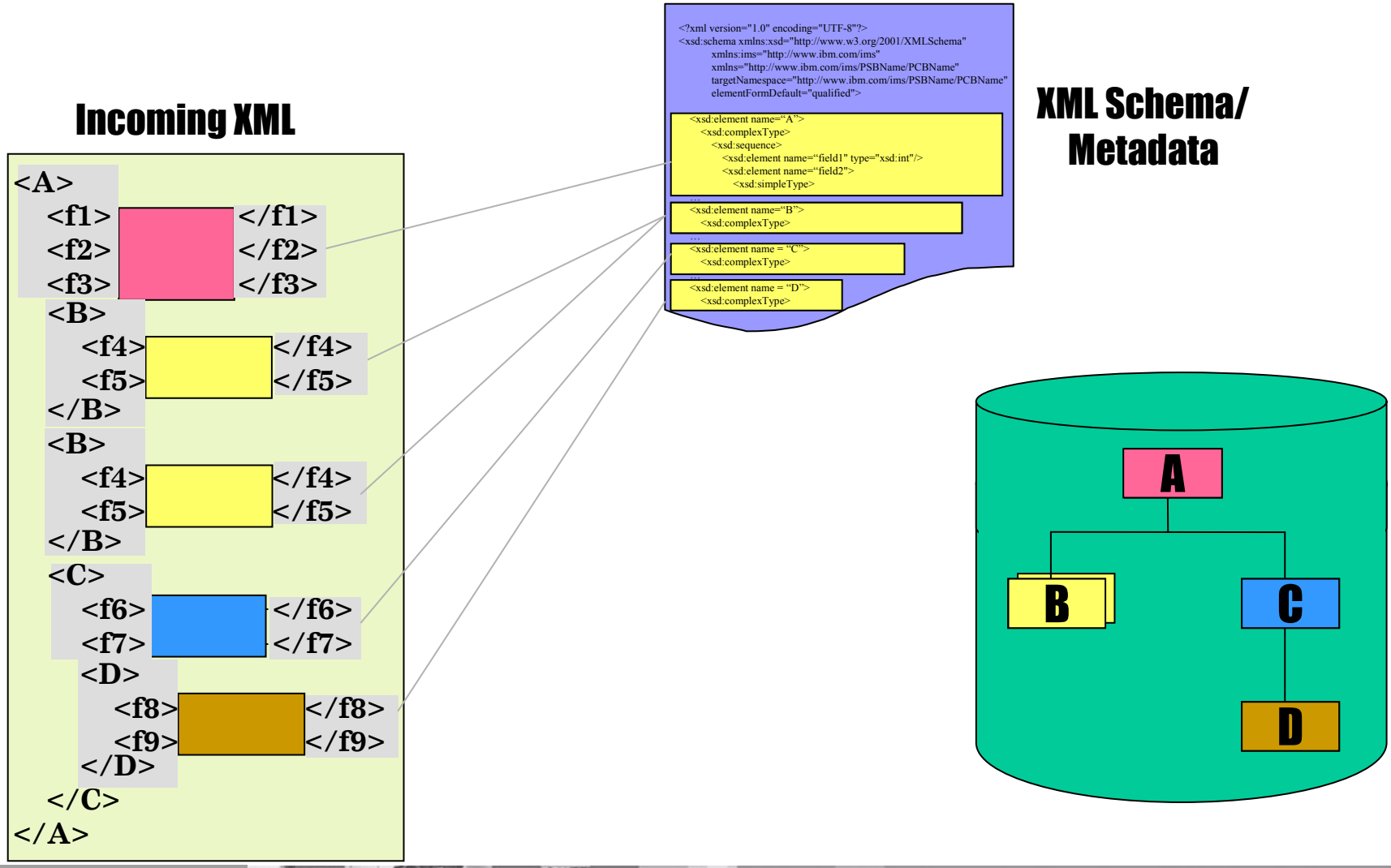
XML Schema/ Metadata



Decomposed Storage

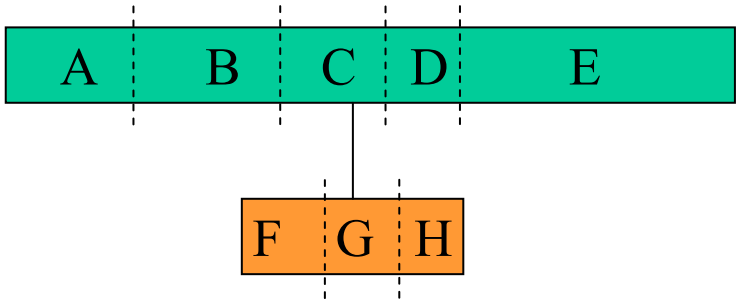
- XML document must be parsed and validated.
- Data must be converted to *traditional* IMS types
  - COMP-1, COMP-2, etc.
  - EBCDIC CHAR, Picture Strings
- Stored data is searchable by IMS and transparently accessible by non-XML enabled applications.

Decomposed XML Storage in IMS



Simple XML Structuring

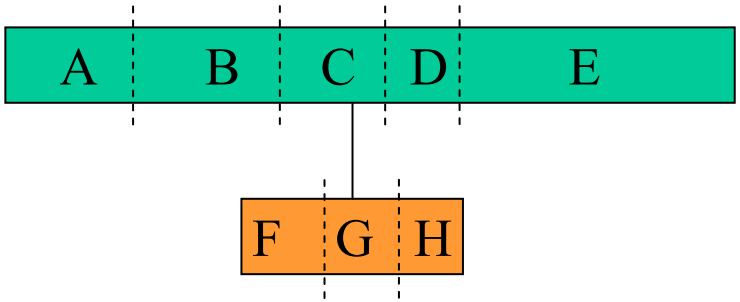
- Uses Default structure
- All data is represented as elements




```
<?xml version="1.0"?>
<SegName>
    <A>      </A>
    <B>      </B>
    <C>      </C>
    <D>      </D>
    <E>      </E>
    <Seg2Name>
        <F>      </F>
        <G>      </G>
        <H>      </H>
    </Seg2Name>
    <Seg2Name>
        <F>      </F>
        <G>      </G>
        …
</SegName>
```

Complex XML Structuring

- Uses Attribute Mapping
- Uses hard coded values
- Uses hard coded levels

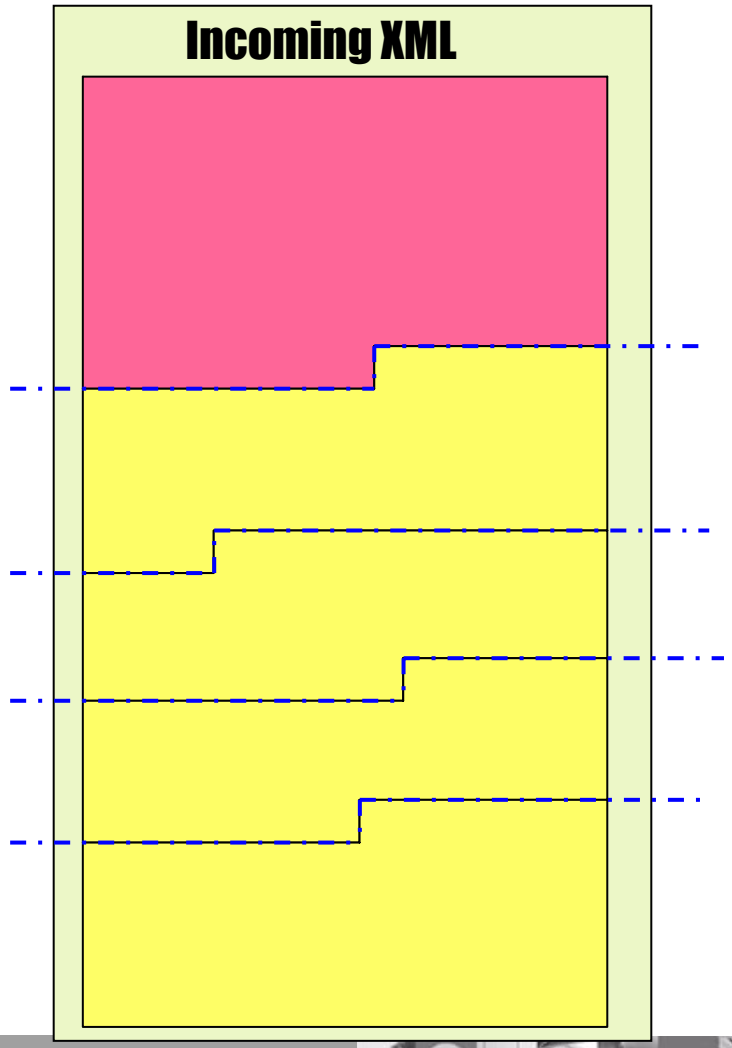


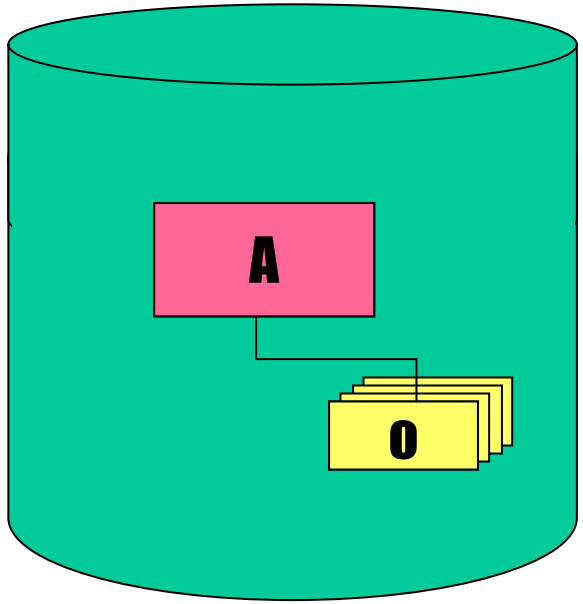

```
<?xml version="1.0"?>
<document A="[ ]" B="[ ]">
    <G F="[ ]" H="[ ]">[ ]</G>
    <G F="[ ]" H="[ ]">[ ]</G>
    …
    <sub C="[ ]">
        <another attr="38">
            <E D="[ ]">[ ]</E>
    </sub>
</document>
```

Intact Storage

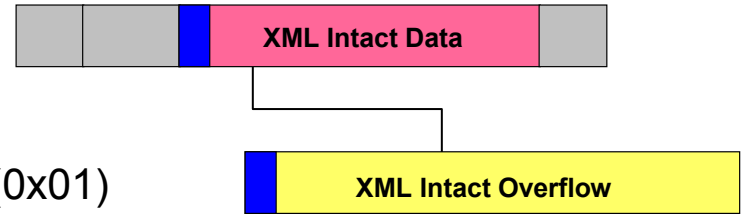
- No (or little) XML Parsing or Schema validation
  - Storage and Retrieval Performance
- No (or little) data type conversions
  - Unicode storage
- Stored documents are no longer searchable by IMS and only accessible to XML-enabled applications
  - XPath side segments

Intact XML Storage in IMS





Intact Storage Structure

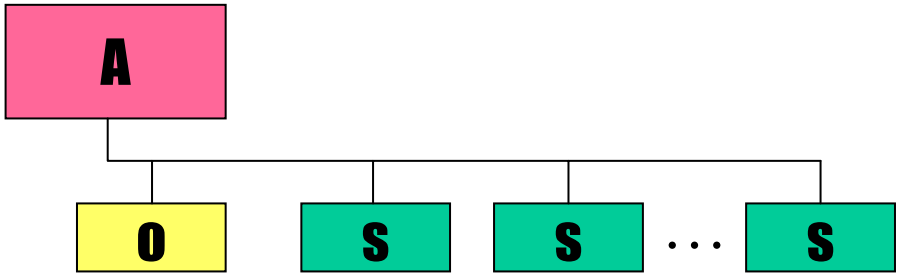


- Base Segment
  - 1 byte — Version Number (0x01)
  - 1 byte — Reserved
  - 2 bytes — 1st bit: indicates more segments than just this one
    2nd–16th bits: indicates this segments length
  - n bytes — Intact XML Data

- Overflow Segment
  - 2 bytes — Key field sequence number
  - 2 bytes — 1st bit: indicates more segments than just this one
    2nd–16th bits: indicates this segments length
  - n bytes — Intact XML Data

Intact Storage Secondary Indexing

- XPath expression identifying Side Segments
  - Side segment is converted to *traditional* data type and copied into segment.
- Side Segments are secondary indexed with documents root as target.

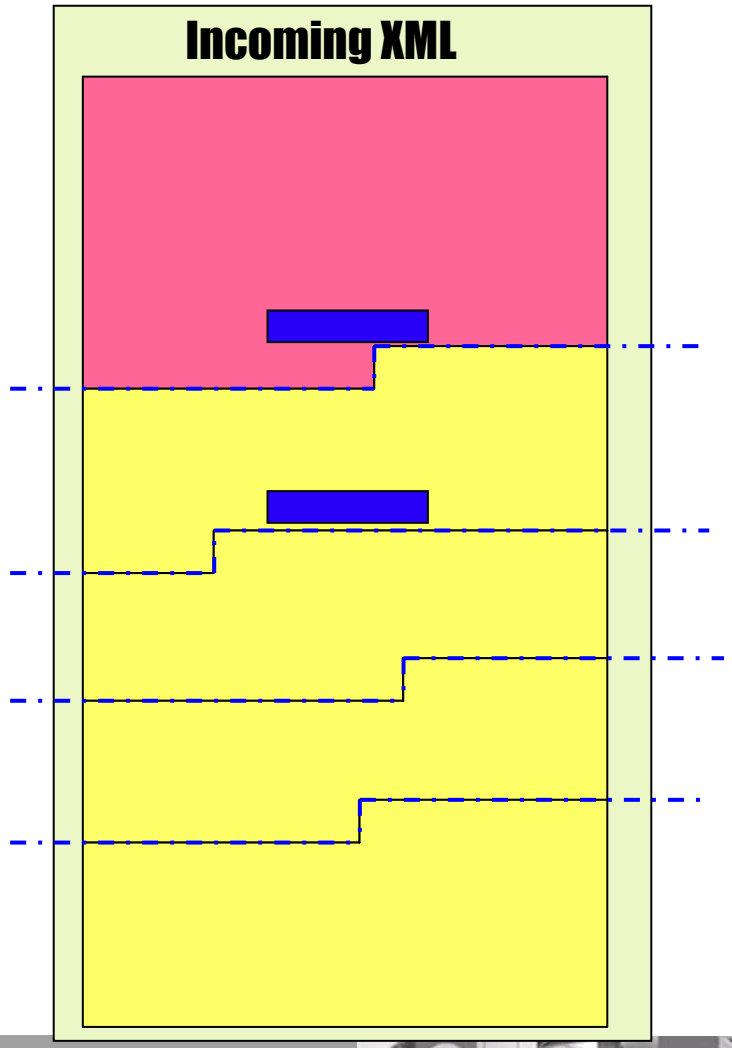


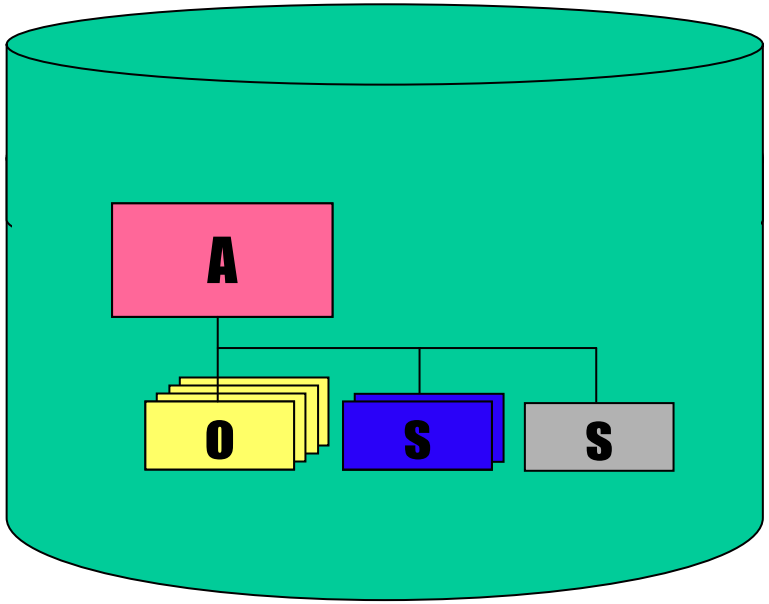
Example:

XPath="/Dealer/DealerName"

XPath="/Dealer/Model[Year>1995]/Order/LastName"

Intact XML Storage in IMS





Example:

XPath="/A/B/f4"
XPath="/A/E/f1"

Decomposed vs. Intact Storage

- Decomposed (*data-centric storage*)
  - XML tags are stripped from XML data
  - Identical as current IMS storage
  - Strict data-centric XML Schema validated data
  - EBCDIC encoding
  - Searching on IMS Search Fields

- Intact (*document-centric storage*)
  - Entire XML document is stored (including tags)
  - Relaxed un-validated data
  - Any desired encoding is possible
  - Searching is through XPath specified and generated Secondary Indexed Side Segments

Overview

- Introduction
  - What is IMS XML-DB?
  - What is XML?
  - What are XML Schemas?
- IMS XML-DB Methodology (XML Visualization)
  - Decomposed Storage
  - Intact Storage
- IMS Java / XML-DB Tooling
  - Metadata Generation
- IMS XML-DB Java Implementation
  - SQL UDF Interface
  - Future Direction

DL/I Model Utility

Control statements:

1) **Choose PSBs/DBDs**
2) **Choose copybook members**
3) **Aliases, data types, new fields.**

If you can read this you do not need glasses; however this is just silly writting to represent the control statements that are the input to the utility.

COBOL copybook members

(HFS or PDS)

PSB

DBD

(PDS)

DL/I Model Utility

XMI 1.2 (HFS)

XML Schema(s) (HFS)

IMS Java Metadata classes (HFS)

IMS Java Report (HFS)

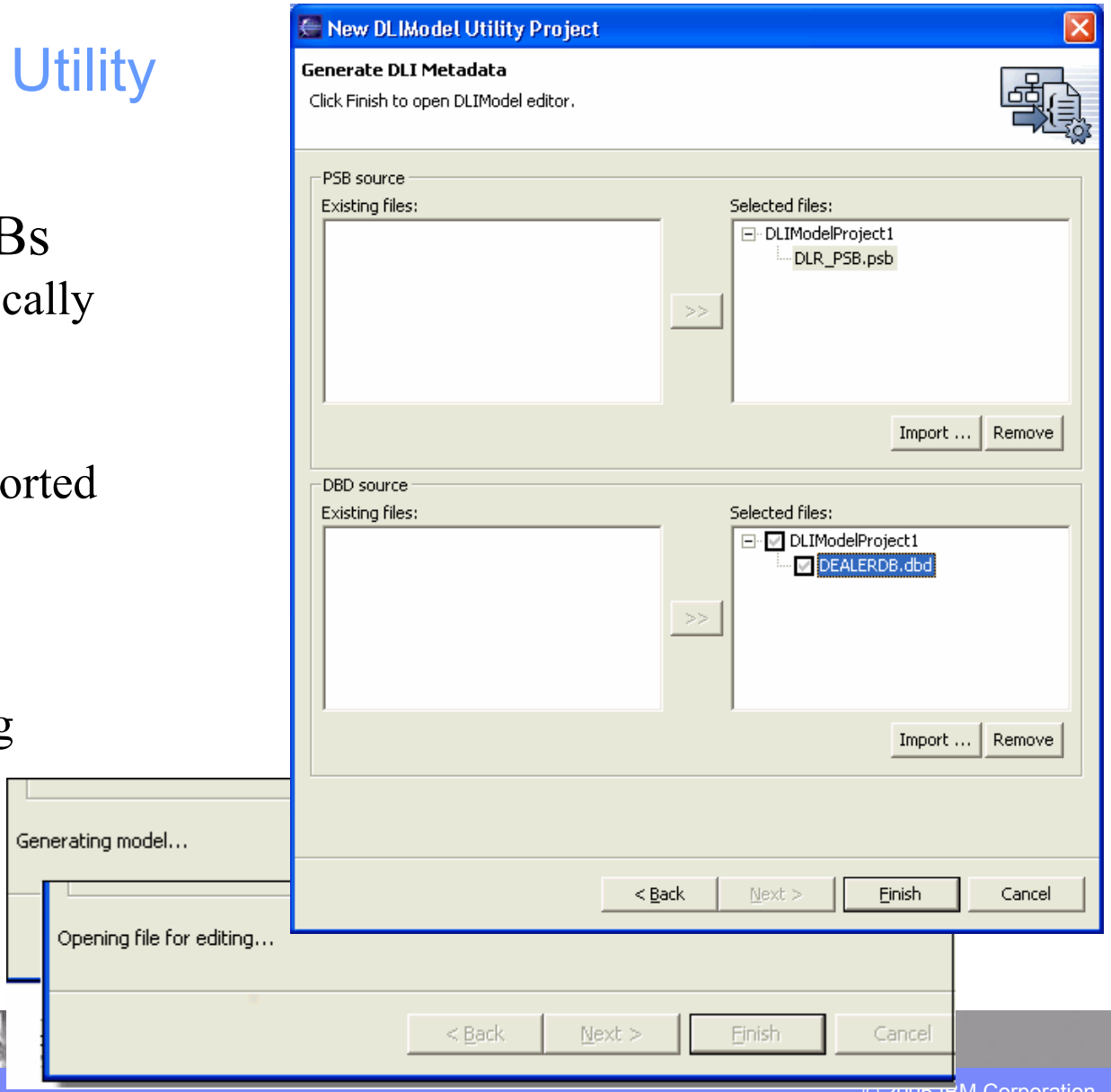
DL/I Model Schema Generation

- Additional Control Statements Keywords

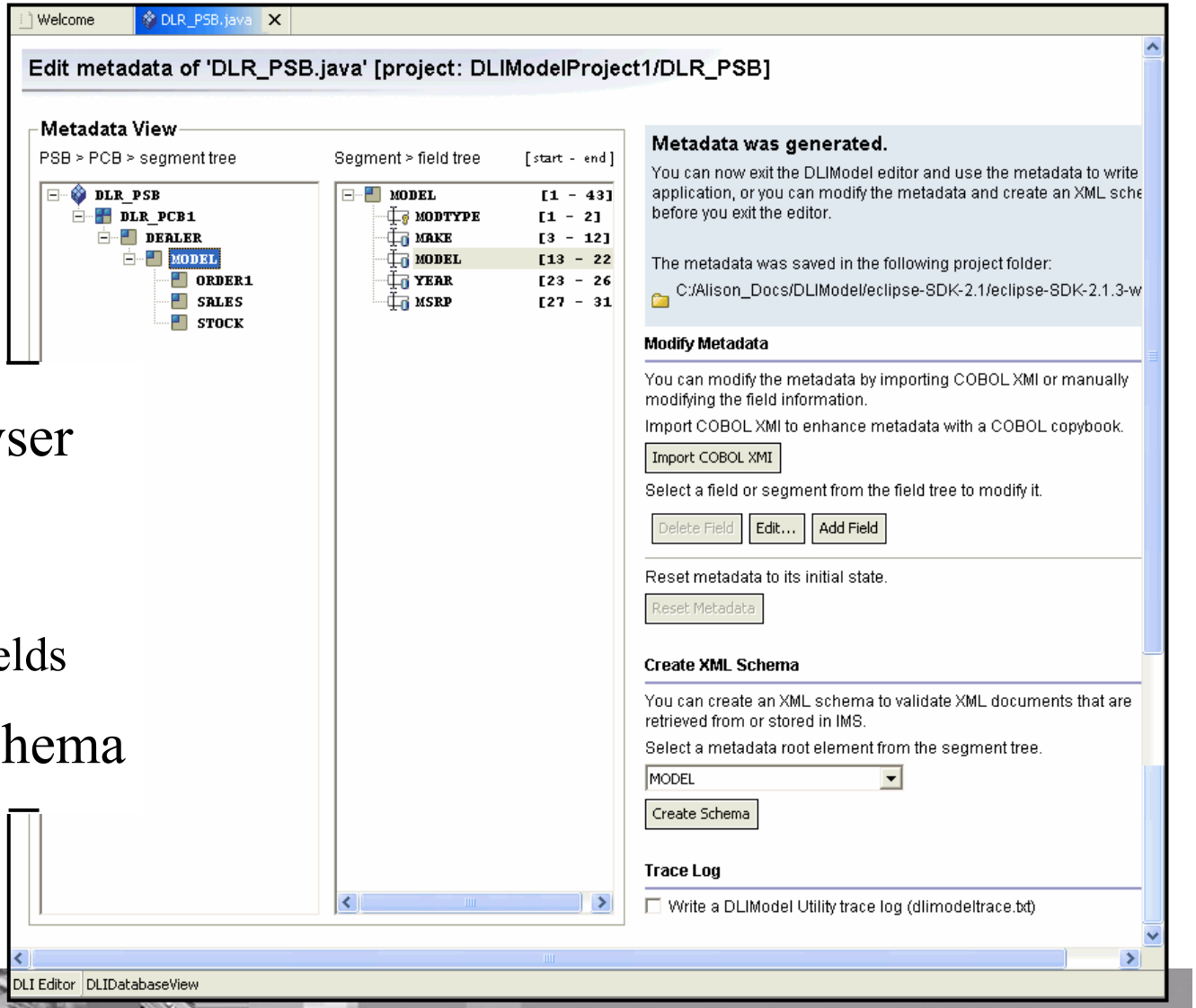
```
OPTIONS PSBds=PSB.SOURCE.PDS     DBDds=DBD.SOURCE.PDS
        GenJavaSource=YES        JavaSourcePath=output/dir
        Package=test.db.psb4     ReportPath=output/dir
        GenXMLSchemas=store      XMLSchemaPath=output/dir
        Outpath=output/dir

PSB psbName=PREPSB4 Javaname=PresentationDatabase
    PCB PCBName=PCB1 JavaName=FullView  GenXMLSchema=store

// Physical Segments for PREZDB
SEGM DBDName=PREZDB SegmentName=PRSNTON JavaName=Presentation
    FIELD Name=TITLE      JavaType=CHAR      JavaName=title
    FIELD Name=LENGTH JavaType=INTEGER  JavaName=length
            …
            …
```

GUI DL/I Model Utility

- Select DBDs / PSBs
  - must be FTPed locally
- Source is Parsed
  - any errors are reported
- XMI Metamodel
  - generated
  - opened for editing

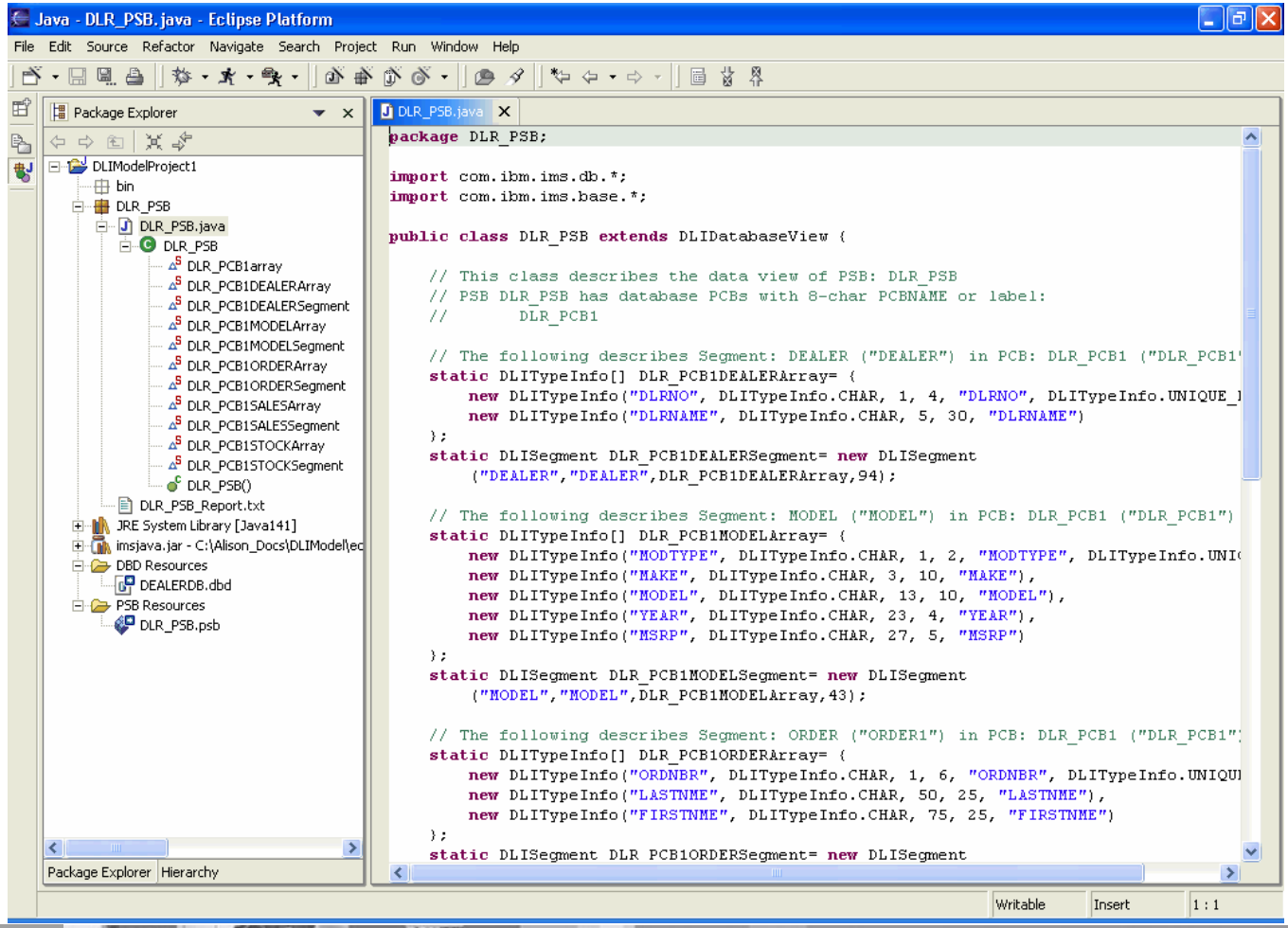


DL/I Model GUI Editor

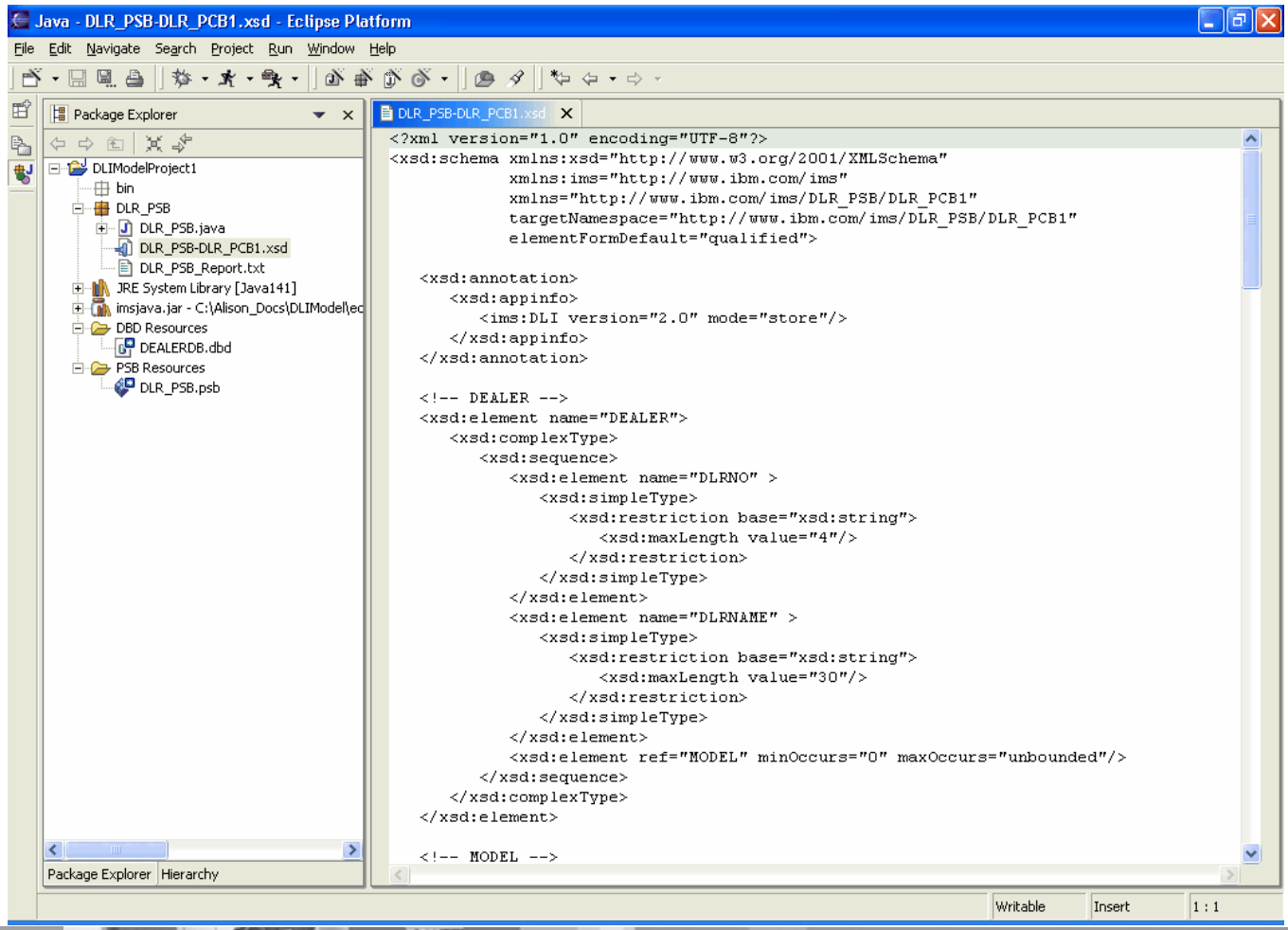


- Hierarchical Browser
- Segment View
  - Import COBOL
  - Add/Edit User Fields
- Generate XML Schema

GUI DL/I Model Utility – Physical Metadata

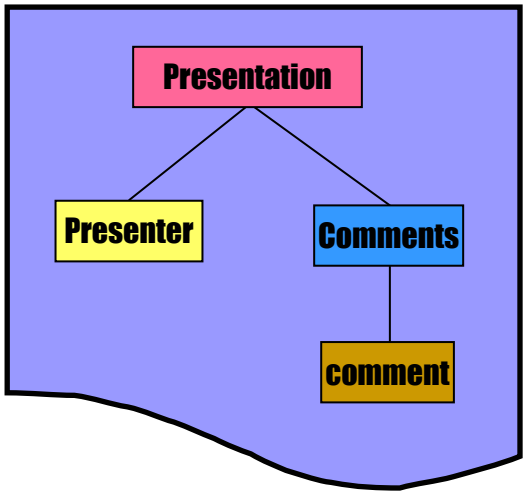


GUI DL/I Model Utility – Structural Metadata

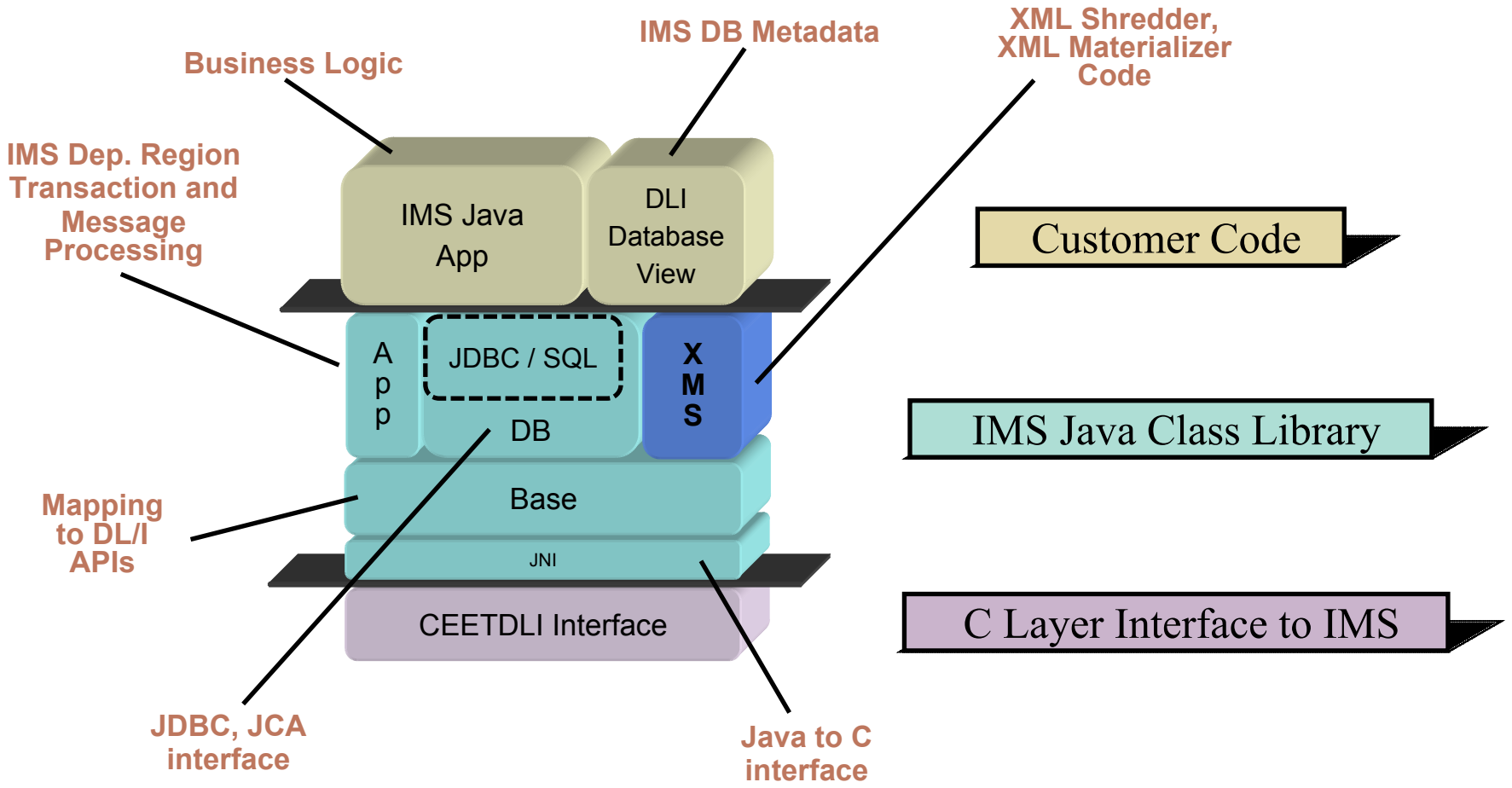


XML Schema Example

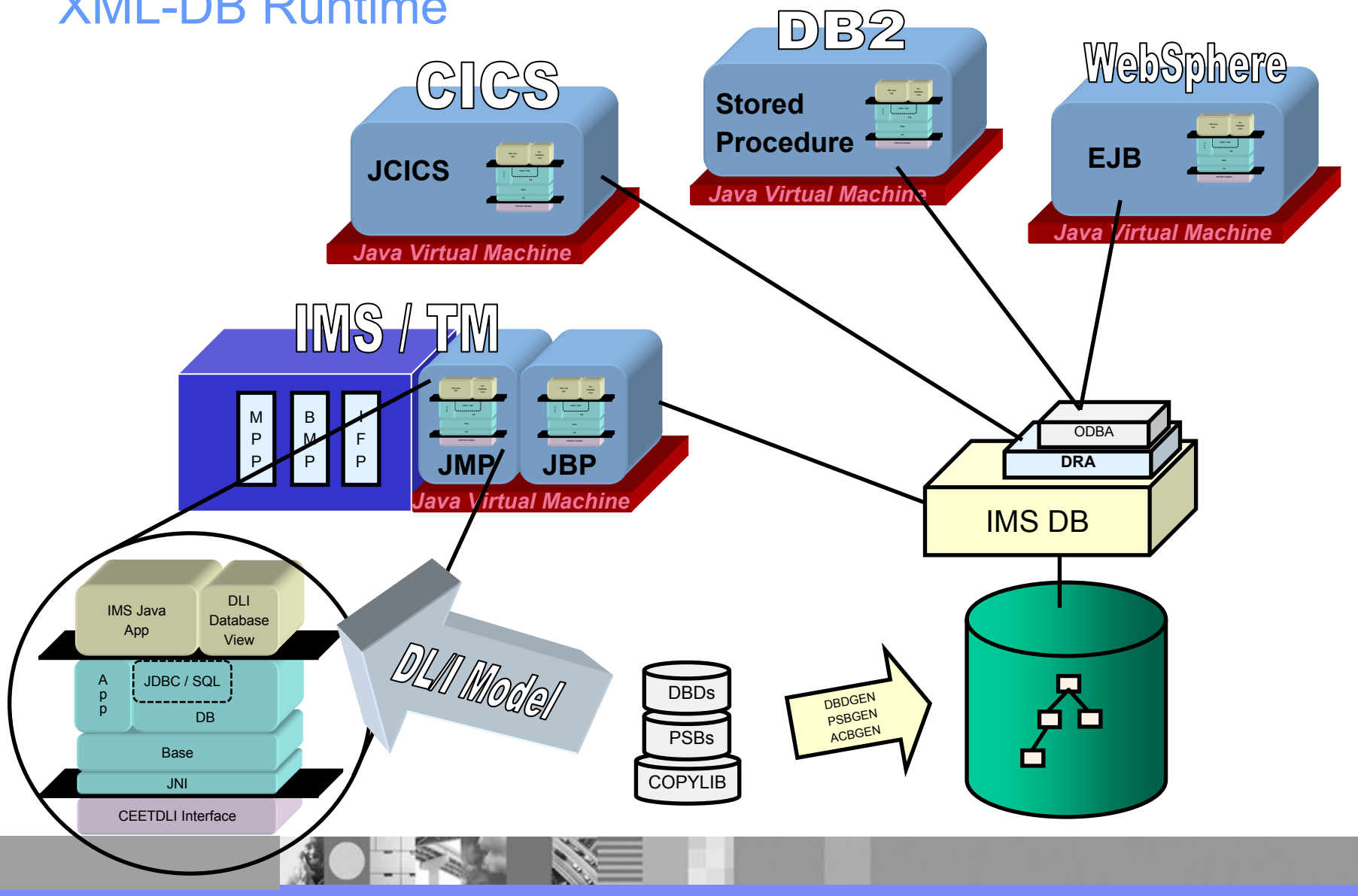
```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
        xmlns="http://www.ibm.com/ims/PREPSB4/FullView"
        targetNamespace="http://www.ibm.com/ims/PREPSB4/FullView"
        elementFormDefault="qualified">

 <xsd:annotation>
    <xsd:appinfo>
      <ims:DLI mode="store"/>
    </xsd:appinfo>
  </xsd:annotation>

   <xsd:element name="Presentation">
     <xsd:complexType>
       <xsd:sequence>
          <xsd:element name="title" type="xsd:string"/>
          <xsd:element name="length" type="xsd:integer"/>
          <xsd:element ref="Presenter" minOccurs="0" maxOccurs="unbounded"/>
          <xsd:element ref="Comments" minOccurs="0" maxOccurs="unbounded"/>
    …

</xsd:schema>
```



IMS XML-DB Java Implementation



XML-DB Runtime

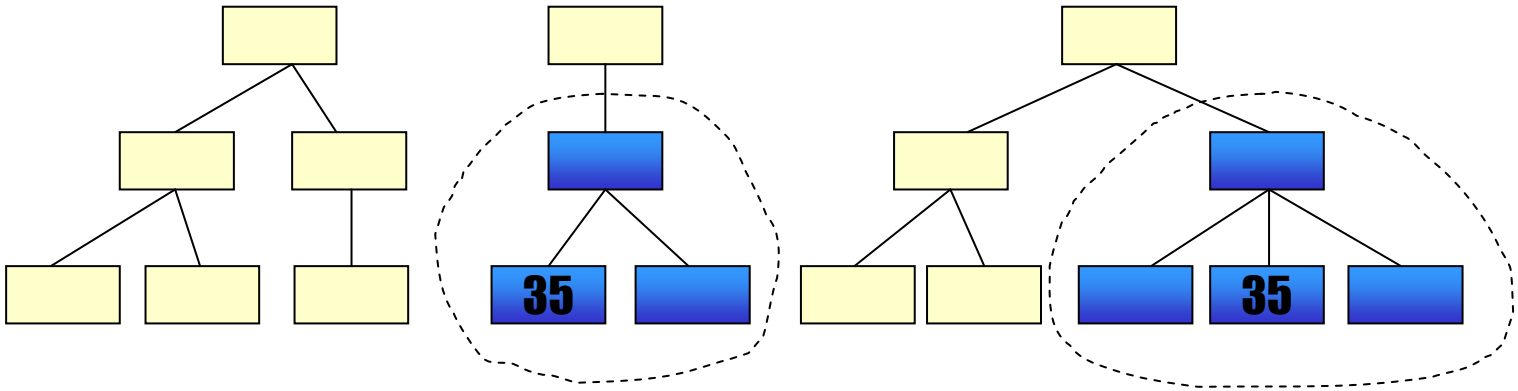


IMS XML-DB Java Interface

- Adds 2 User Defined Funtions (UDF) to the IMS Java JDBC SQL interface
  - retrieveXML()
  - storeXML()
- Runs as an IMS Java Application
  - JDR (JMP, JBP) *(IMS TM 9.1 or higher)*
  - DB2 Stored Procedure *(DB2 v7 or higher)*
  - CICS *(CICS TS 2.2 or higher)*
  - WebSphere *(5.01 or higher)*

RetrieveXML() UDF

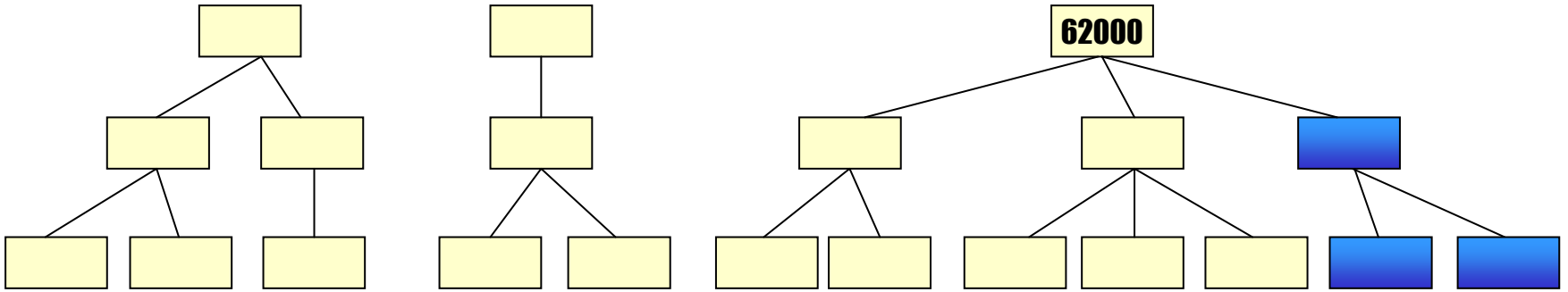
```
SELECT retrieveXML(B)
FROM C
WHERE C.fieldA = ’35’
```



**Two Rows of XML CLOBs in the ResultSet*

StoreXML() UDF

```
INSERT INTO B (storeXML())
VALUES (?)
WHERE A.fieldA = ’62000’
```



**Insert Statement must be a Prepared Statement*

Execute Query

retrieveXML() call

```
public void processMessage(String dealerName) {
    obtain connection...

    String query =
        "SELECT DealerSegment.DealerName, retrieveXML(DealerSegment) AS DealerXMLDoc" +
        " FROM Dealer.DealerSegment" +
        " WHERE DealerSegment.DealerName = '" + dealerName + "'";

    Statement statement = connection.createStatement();
    ResultSet results = statement.executeQuery(query);

    process results...
    close connection...
}
```

Process Results

getClob() call

```
public void processMessage(String dealerName) {
     obtain connection...
     execute query...

     while (results.next()) {

          Clob xmlDoc = results.getClob("DealerXMLDoc");

          saveClobToFile(xmlDoc, results.getString("DealerName"));
     }

     close connection...
 }
```

Process Results

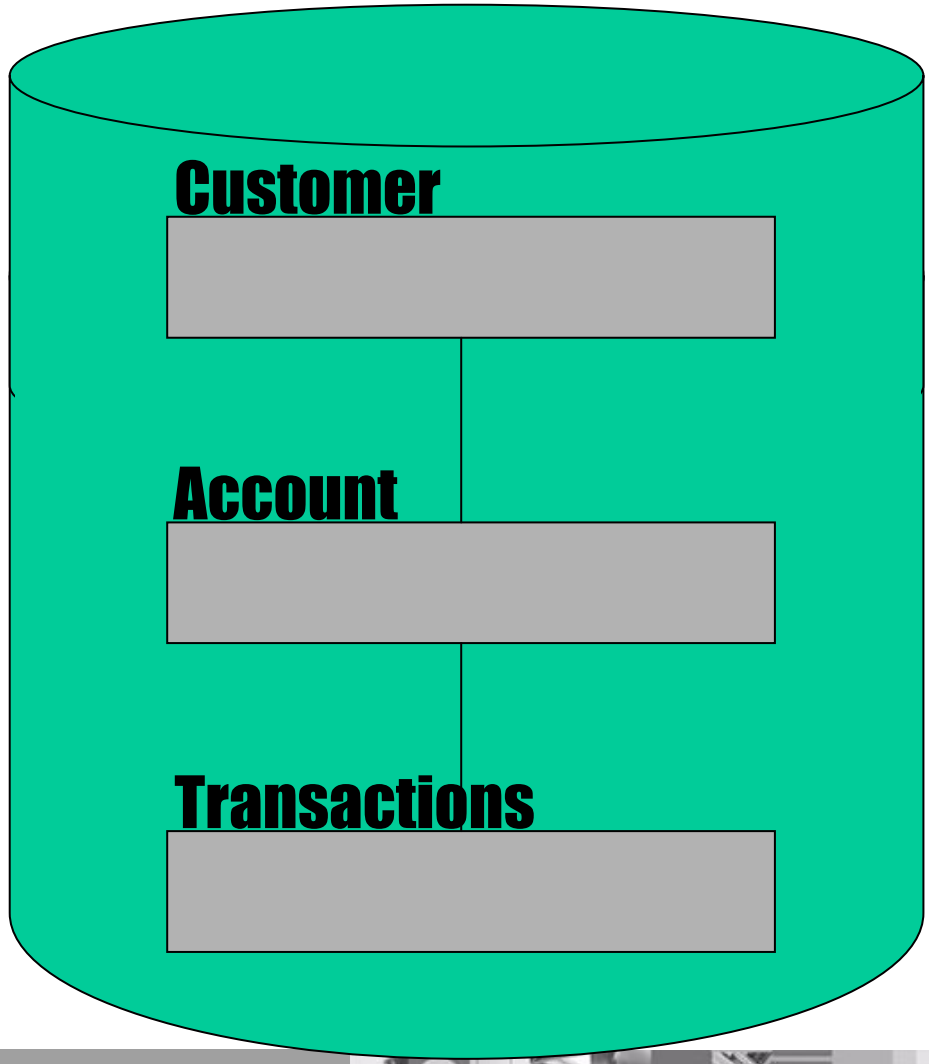
getCharacterStream() or getAsciiStream()

```
public void saveClobToFile(Clob clob, String fileName) throws IOException {

    Reader reader = clob.getCharacterStream();
    FileWriter writer = new FileWriter(fileName + ".xml");

    char[] line = new char[1024];
    int x = reader.read(line,0,1024);
    while (x != -1) {
        writer.write(line,0,x);
        x = reader.read(line,0,1024);
    }

    reader.close();
    writer.close();
}
```

IMS XML-DB Interface Future

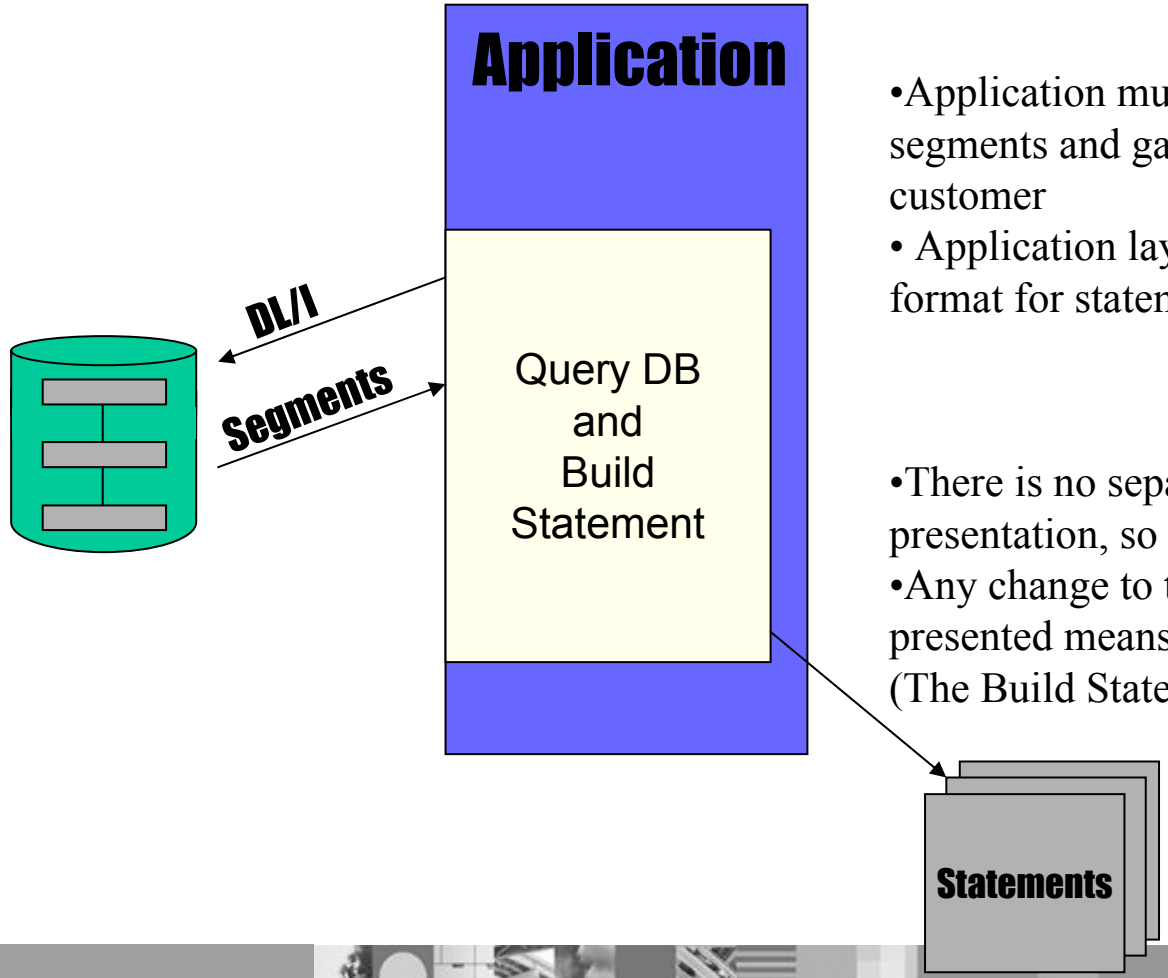
- SQL is a questionable IMS/XML interface
  - Hierarchical DB
  - Hierarchical Data
  - Relational Query Language??
- SQL/XML
  - Still relational
- XPath / XQuery
  - Only query right now
  - Still under development

Hypothetical Bank DB



- Every month send customer statements
- On-line Account access
- etc.

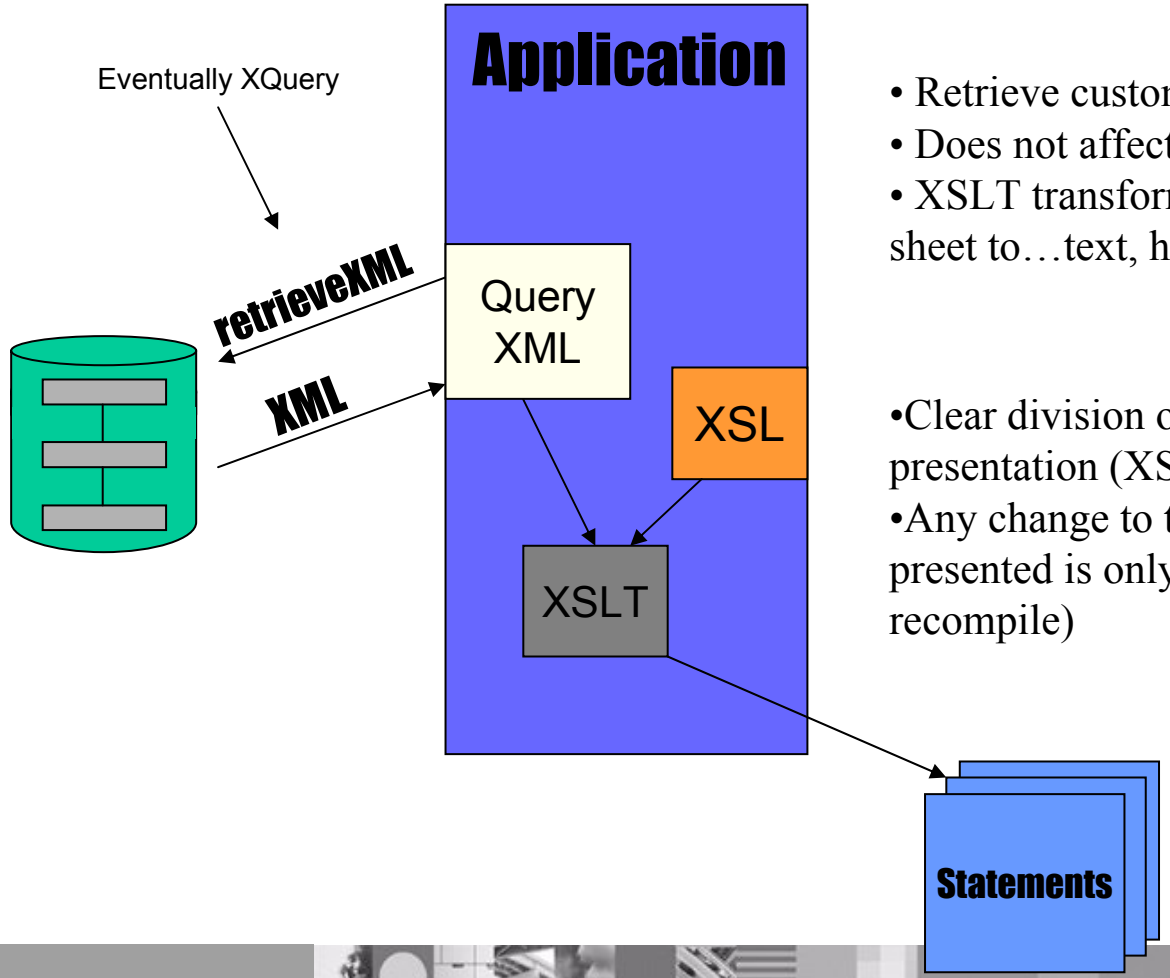
Current tedious design



- Application must query all needed DL/I segments and gather needed data for each customer
- Application lays out data in desired output format for statement or web page.

- There is no separation of data and presentation, so
- Any change to the way the data is to be presented means a change to the application (The Build Statement Module).

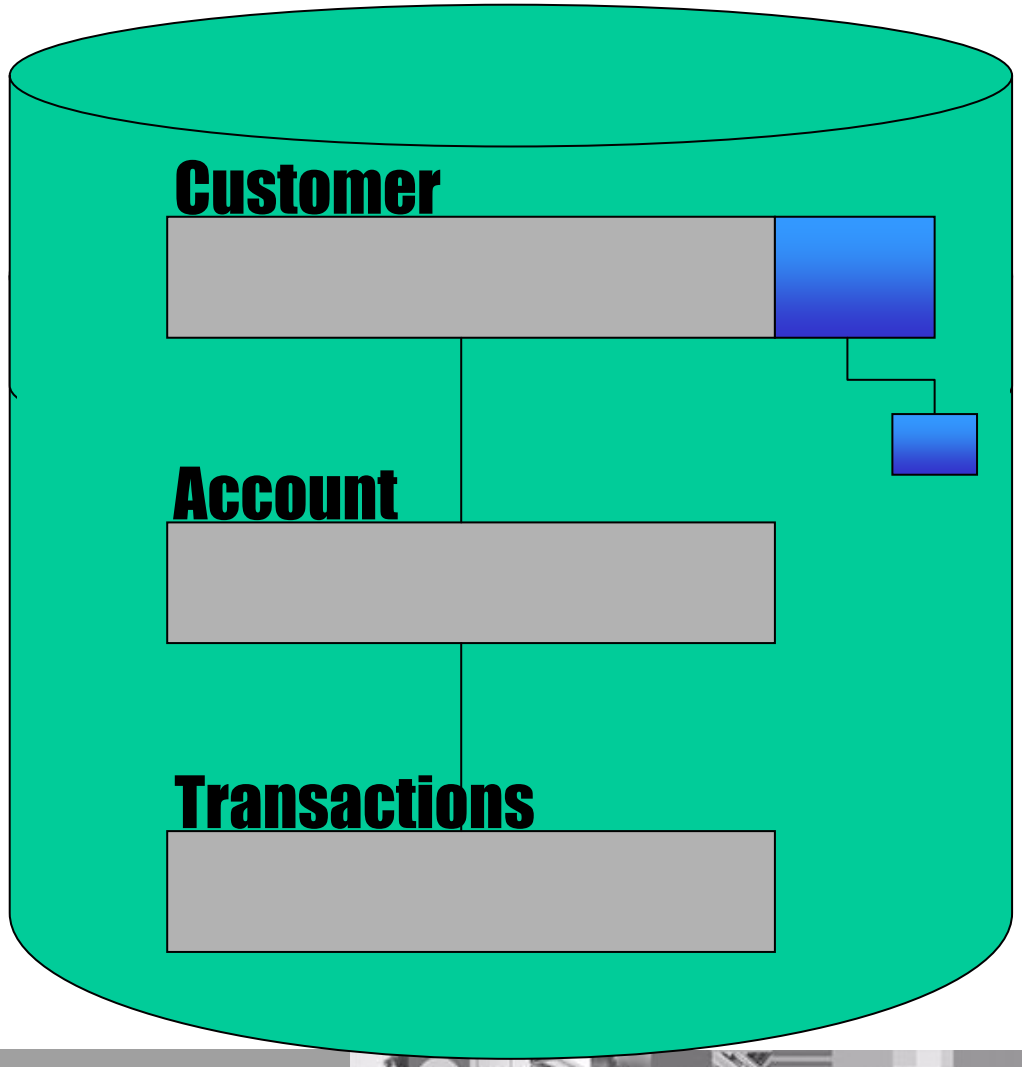
New XML design



- Retrieve customer data in XML format
- Does not affect other apps
- XSLT transforms XML based on style sheet to…text, html, PDF, etc.

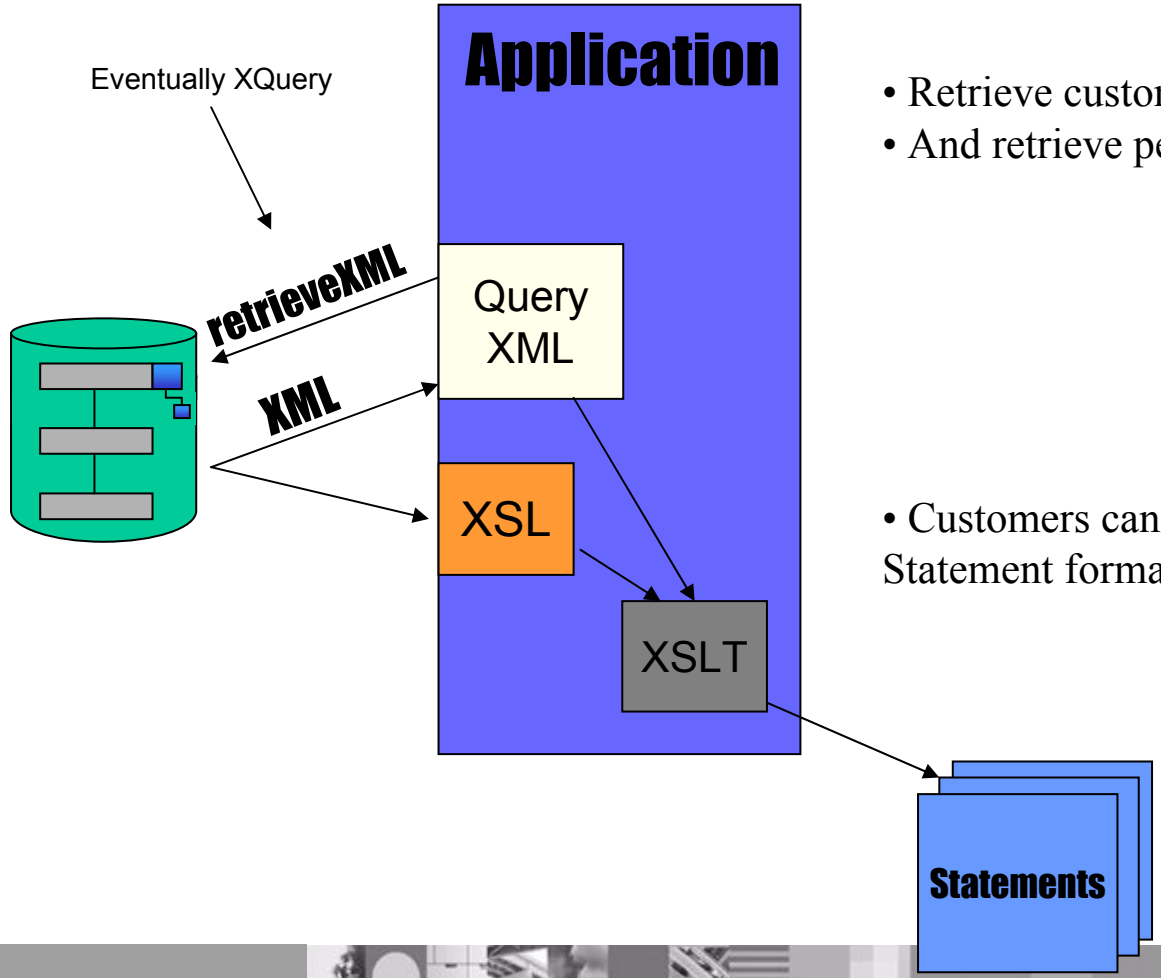
- Clear division of data (XML from DB) and presentation (XSL)
- Any change to the way the data is to be presented is only change to style sheet (no recompile)

Extended DB with Intact Storage



- Extend Customer with intact XML for each customers own personalized style sheet

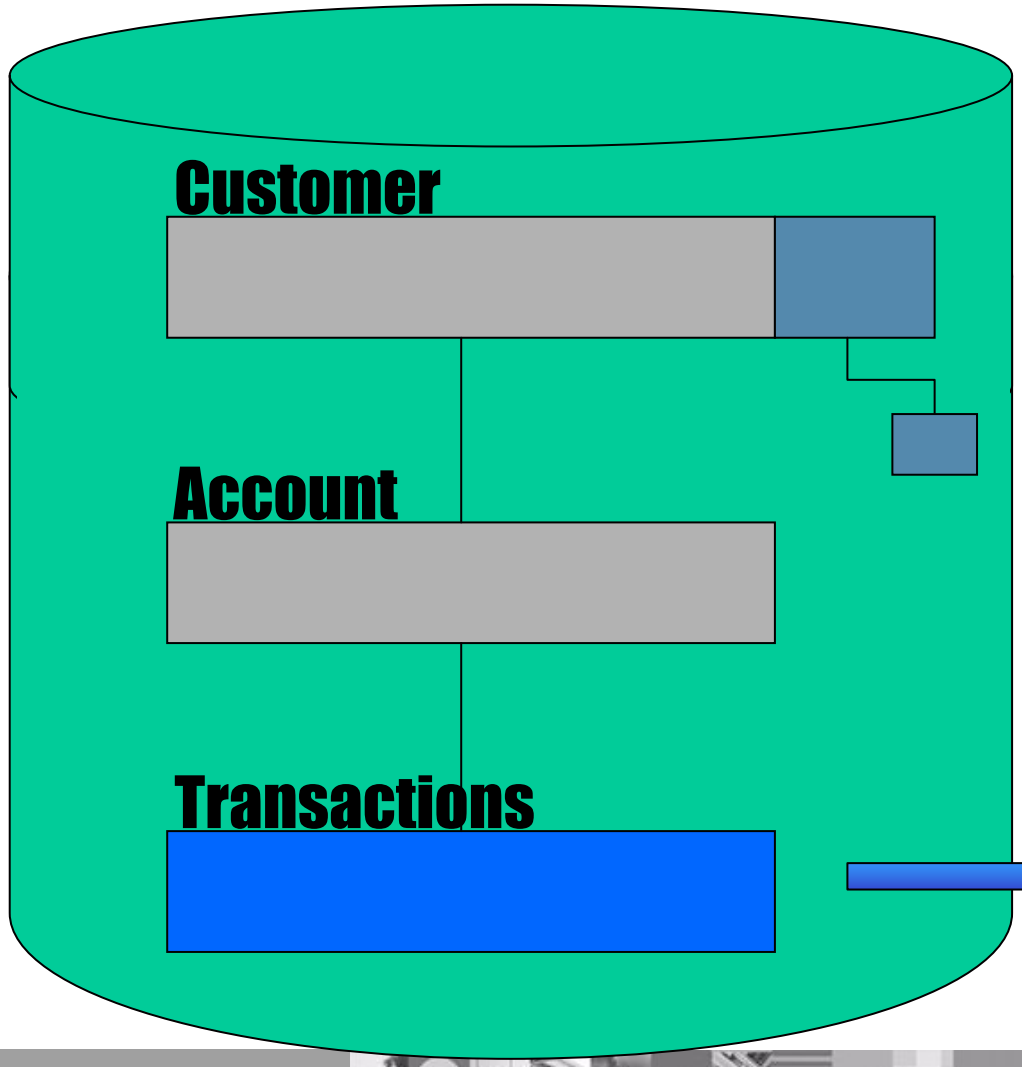
New XML design



- Retrieve customer data in XML format
- And retrieve personalized XSL

- Customers can change their own Bank Statement format

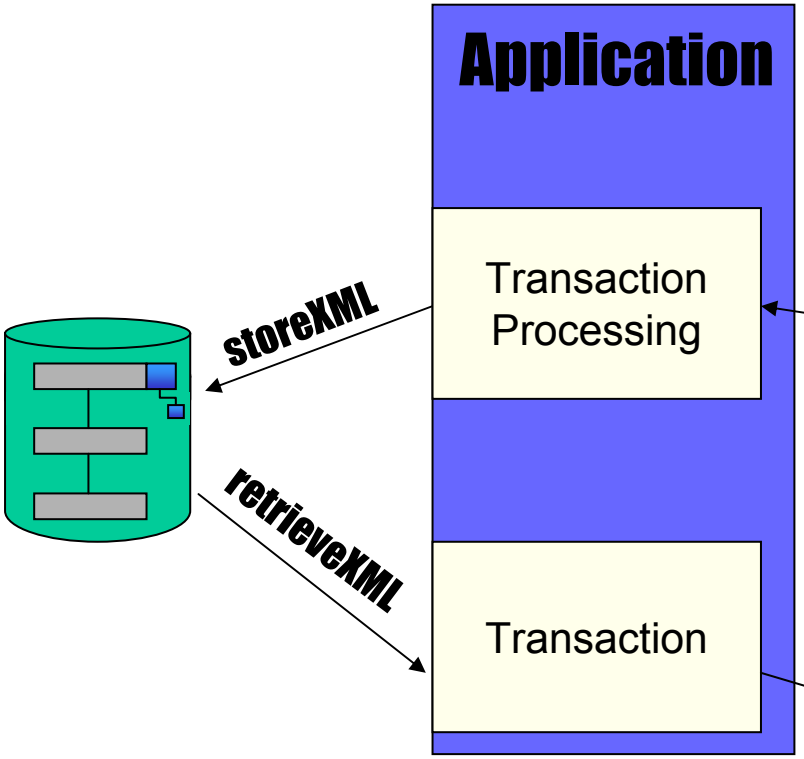
Possible further extension



- Generate Schema for transactions and distribute to suppliers.

XML Schema

New XML storage and retrieval



- Transactions come in as XML (SOAP)
- Normal Tran processing
- Store Transaction directly into database.
- Other apps do not need to change.

XML

- Full document easily retrieved or searched

XML

GUI DL/I Model Utility – Future